Working with XcodeKit

Aryaman Sharda

SwiftLeeds

👋 Hi, I’m Aryaman

@aryamansharda

- 👨‍💻 Staff iOS Engineer at **Turo**
- 👨‍💻 Previously iOS & CarPlay Engineer **at Porsche**
- ✍️ Blog at **digitalbunker.dev**
- ✉️ Curator of the **indie.watch** newsletter
- 🌉 Based in **San Francisco**

What We’ll Cover

- 📚 **Overview**
- 🚀 Building Editor Extensions
- 💥 Breaking The Rules
- 📦 Distribution
- 📥 Installation

What are Source Editor Extensions?

XcodeKit

- **Add custom commands** to Xcode’s Editor menu
- Modify the **current source** file
- Edit the **user’s text selection**
- **Navigate** within the source file
- Extensions run in a **separate process** so Xcode stays safe and “stable”
- Build tools that **simplify your workflow**

It’s cooler than it sounds, I promise 🤞

EditKit

Easy JSON Formatting & Create Codable Models

Xcode
File
Edit
View
Find
Navigate
Editor
Product
Debug
Integrate
Window
Help
Sat Sep 30 8:45 PM
SwiftLeeds
main
SwiftLeedsDemo
My Mac
Build Succeeded | 9/27/23 at 12:42 PM
SourceEditorExtension.swift
SourceEditorCommand.swift
ContentView.swift
CustomerLocation.swift
SwiftLeedsApp.swift
SwiftLeeds
SwiftLeeds
CustomerLocation.swift
No Selection
Inferior
Line: 10 Col: 1

```swift
//
//  CustomerLocation.swift
//  SwiftLeeds
//
//  Created by Aryaman Sharda on 9/30/23.
//

import Foundation


```

Xcode File Edit View Find Navigate Editor Product Debug Integrate Window Help
Sat Sep 30 8:45 PM
SwiftLeeds main
SwiftLeedsDemo ❯ My Mac
Build Succeeded | 9/27/23 at 12:42 PM
10
1
SourceEditorExtension.swift
SourceEditorCommand.swift
ContentView.swift
CustomerLocation.swift
SwiftLeedsApp.swift
SwiftLeeds ❯ SwiftLeeds ❯ CustomerLocation.swift ❯ No Selection
Inferior
Line: 25 Col: 17

```
//
//  CustomerLocation.swift
//  SwiftLeeds
//
//  Created by Aryaman Sharda on 9/30/23.
//

import Foundation

[                                                             2 ⚠ ⊗ Expressions are not allowed at the top level
  {
    "secondary_address" : "Apt.633",                          2 ⊙ Consecutive statements on a line must be separated by ';'
    "zip" : "11449-3132",
    "city_prefix" : "West",
    "city_suffix" : "mouth",
    "street_suffix" : "Points",
    "country" : "AmericanSamoa",
    "country_code" : "GF",
    "full_address" : "46437DwightManor,Pfannerstilltown,NV34784",
    "street_address" : "34244LarkinJunctions",
    "latitude" : 49.913237499767575,
    "city" : "Mertzland",
    "state" : "Alaska",
    "community" : "ParadiseCrossing",
    "id" : 2135,
    "mail_box" : "POBox33",
    "zip_code" : "97835",
    "uid" : "18043204-8fd1-43d7-b2a5-b36d42b34f9d",
    "building_number" : "26348",
    "longitude" : 5.776890360671274,
    "street_name" : "VeumTerrace",
    "time_zone" : "Pacific\/Port_Moresby",                   ⊗ Invalid escape sequence in literal
    "state_abbr" : "VA",
    "postcode" : "58068-2817"
  },
  {
    "secondary_address" : "Apt.736",                          2 ⊙ Consecutive statements on a line must be separated by ';'
    "zip" : "07160-3241",
```

EditKit

Quickly Localize Text

ScreenFlow File Edit Mark Clip Insert Actions Font Arrange View Window Help
Sun Oct 8 8:53 PM
SwiftLeedsUIKit
main
SwiftLeedsUIKit › iPhone 15 Pro
SwiftLeedsUIKit: Ready | Today at 8:53 PM
ViewController.swift
SwiftLeedsUIKit › SwiftLeedsUIKit › ViewController.swift › viewDidLoad()
SwiftLeedsUIKit
SwiftLeedsUIKit
AppDelegate.swift
SceneDelegate.swift
ViewController.swift M
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
Filter
Line: 14 Col: 24

```swift
//
//  ViewController.swift
//  SwiftLeedsUIKit
//
//  Created by Aryaman Sharda on 10/8/23.
//

import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        title = "Login"
    }
}


```

Swiftify

Objective-C to Swift Converter

Xcode File Edit View Find Navigate Editor Product Debug Integrate Window Help
Sat Sep 30 2:47 PM
WMFAppViewController.m
WMFAppViewController.m › WMFAppTabType
Line: 26 Col: 1

```objc
#import "WMFAppViewController.h"
@import WMF;
@import SystemConfiguration;
#import "Wikipedia-Swift.h"

#define DEBUG_THEMES 1

// Views
#import "UIViewController+WMFStoryboardUtilities.h"
#import "UIApplicationShortcutItem+WMFShortcutItem.h"

// View Controllers
#import "WMFSettingsViewController.h"
#import "WMFFirstRandomViewController.h"

#import "AppDelegate.h"

#import "WMFDailyStatsLoggingFunnel.h"

#import "Wikipedia-Swift.h"
#import "EXTScope.h"

/**
 *  Enums for each tab in the main tab bar.
 */
typedef NS_ENUM(NSUInteger, WMFAppTabType) {
    WMFAppTabTypeMain = 0,
    WMFAppTabTypePlaces = 1,
    WMFAppTabTypeSaved = 2,
    WMFAppTabTypeRecent = 3,
    WMFAppTabTypeSearch = 4
};

/**
 *  Number of tabs in the main tab bar.
 *
 *  @warning Kept as a separate constant to prevent switch statements from being considered inexhaustive. This means we
 *           need to make sure it's manually kept in sync by ensuring:
 *              - The tab enum we increment is the last one
 *              - The first tab enum is initialized to 0
 *
 *  @see WMFAppTabType
 */

static NSTimeInterval const WMFTimeBeforeShowingExploreScreenOnLaunch = 24 * 60 * 60;

static CFTimeInterval const WMFRemoteAppConfigCheckInterval = 3 * 60 * 60;
static NSString *const WMFLastRemoteAppConfigCheckAbsoluteTimeKey = @"WMFLastRemoteAppConfigCheckAbsoluteTimeKey";

static const NSString *kvo_NSUserDefaults_defaultTabType = @"kvo_NSUserDefaults_defaultTabType";
static const NSString *kvo_SavedArticlesFetcher_progress = @"kvo_SavedArticlesFetcher_progress";

```

Copilot For Xcode

Integrates GitHub Copilot & ChatGPT

intitni / CopilotForXcode

Type / to search

Code · Issues 15 · Pull requests 2 · Discussions · Actions · Wiki · Security · Insights

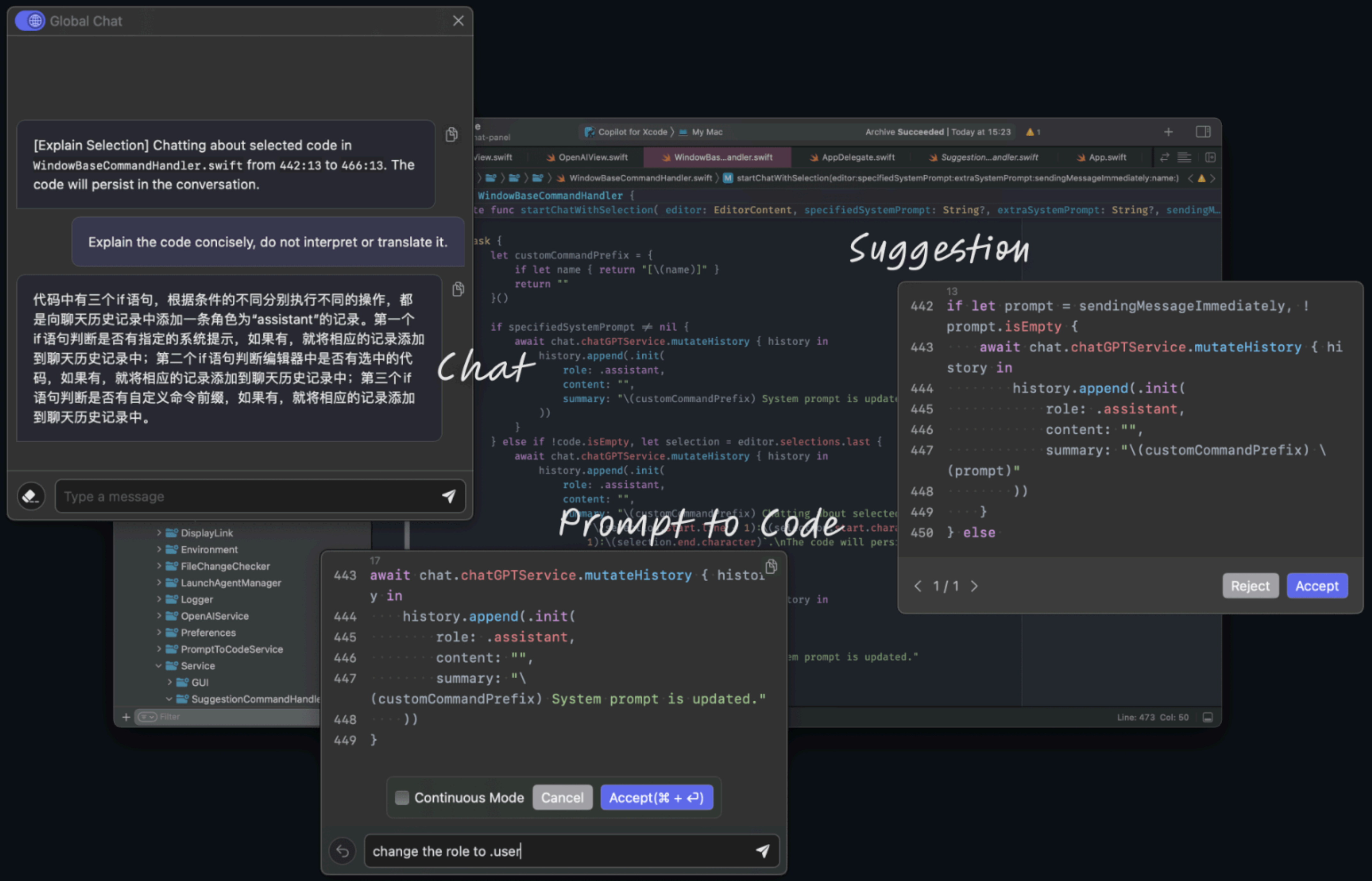
CopilotForXcode Public

Sponsor · Watch 64 · Fork 248 · Star 5.3k

main · 66 branches · 65 tags · Go to file · Add file · Code

README.md

Copilot for Xcode



[Explain Selection] Chatting about selected code in `WindowBaseCommandHandler.swift` from 442:13 to 466:13. The code will persist in the conversation.

Explain the code concisely, do not interpret or translate it.

代码中有三个if语句，根据条件的不同分别执行不同的操作，都是向聊天历史记录中添加一条角色为“assistant”的记录。第一个if语句判断是否有指定的系统提示，如果有，就将相应的记录添加到聊天历史记录中；第二个if语句判断编辑器中是否有选中的代码，如果有，就将相应的记录添加到聊天历史记录中；第三个if语句判断是否有自定义命令前缀，如果有，就将相应的记录添加到聊天历史记录中。

```
if let prompt = sendingMessageImmediately, !prompt.isEmpty {
    await chat.chatGPTService.mutateHistory { history in
        history.append(.init(
            role: .assistant,
            content: "",
            summary: "\(customCommandPrefix) \(prompt)"
        ))
    }
} else
```

```
await chat.chatGPTService.mutateHistory { history in
    history.append(.init(
        role: .assistant,
        content: "",
        summary: "\(customCommandPrefix) System prompt is updated."
    ))
}
```

change the role to .user

Copilot for Xcode is an Xcode Source Editor Extension that provides GitHub Copilot, Codeium and ChatGPT

About

The missing GitHub Copilot, Codeium and ChatGPT Xcode Source Editor Extension

macos · xcode · openai · xcode-extension · copilot · xcode-extensions · github-copilot · githubcopilot · chatgpt · codeium

Readme
View license
Activity
5.3k stars
64 watching
248 forks
Report repository

Releases 59

0.24.1 Latest
4 days ago

+ 58 releases

Sponsor this project

https://intii.lemonsqueezy.com

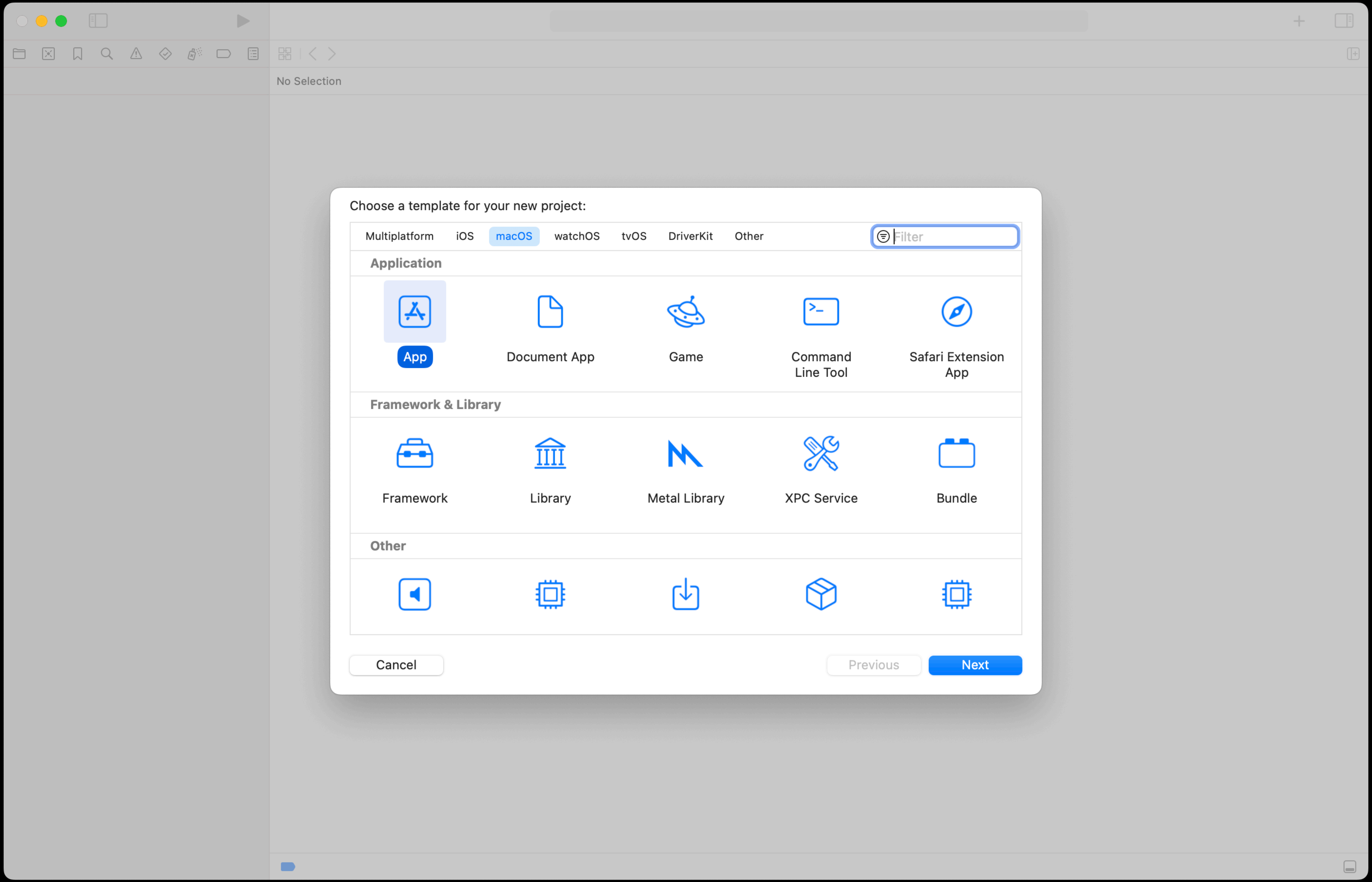
https://www.buymeacoffee.com/intitni

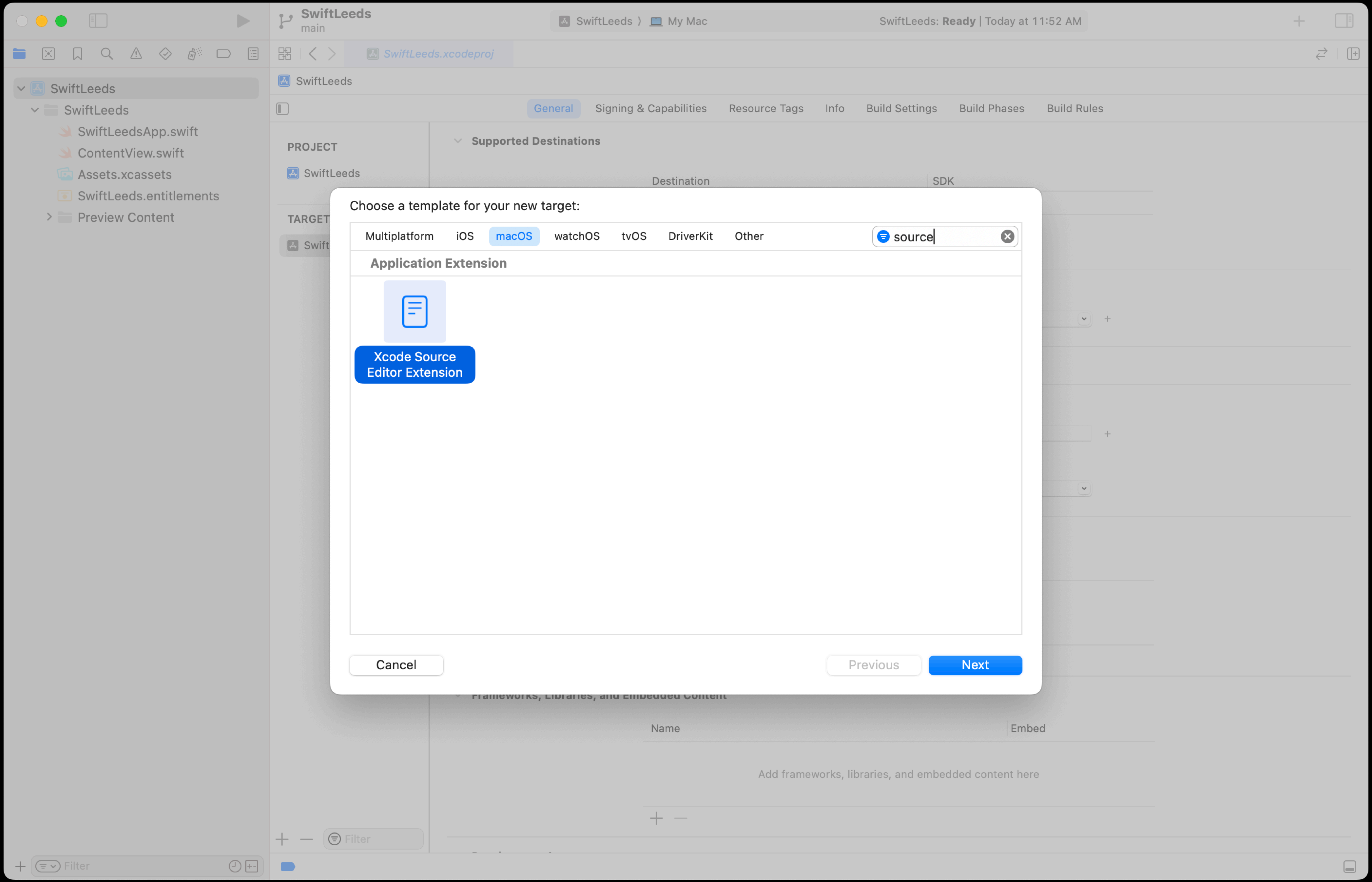
Other Examples

- **Automatically format code** to adhere to style guidelines
- Create **boilerplate code** from user input
- **Faster navigation** with custom commands
- Wrap code in **try-catch blocks, #ifdefs, or other macros**
- **Automatically adding** `// MARK:` comments to code sections
- **Easier refactoring** and documentation

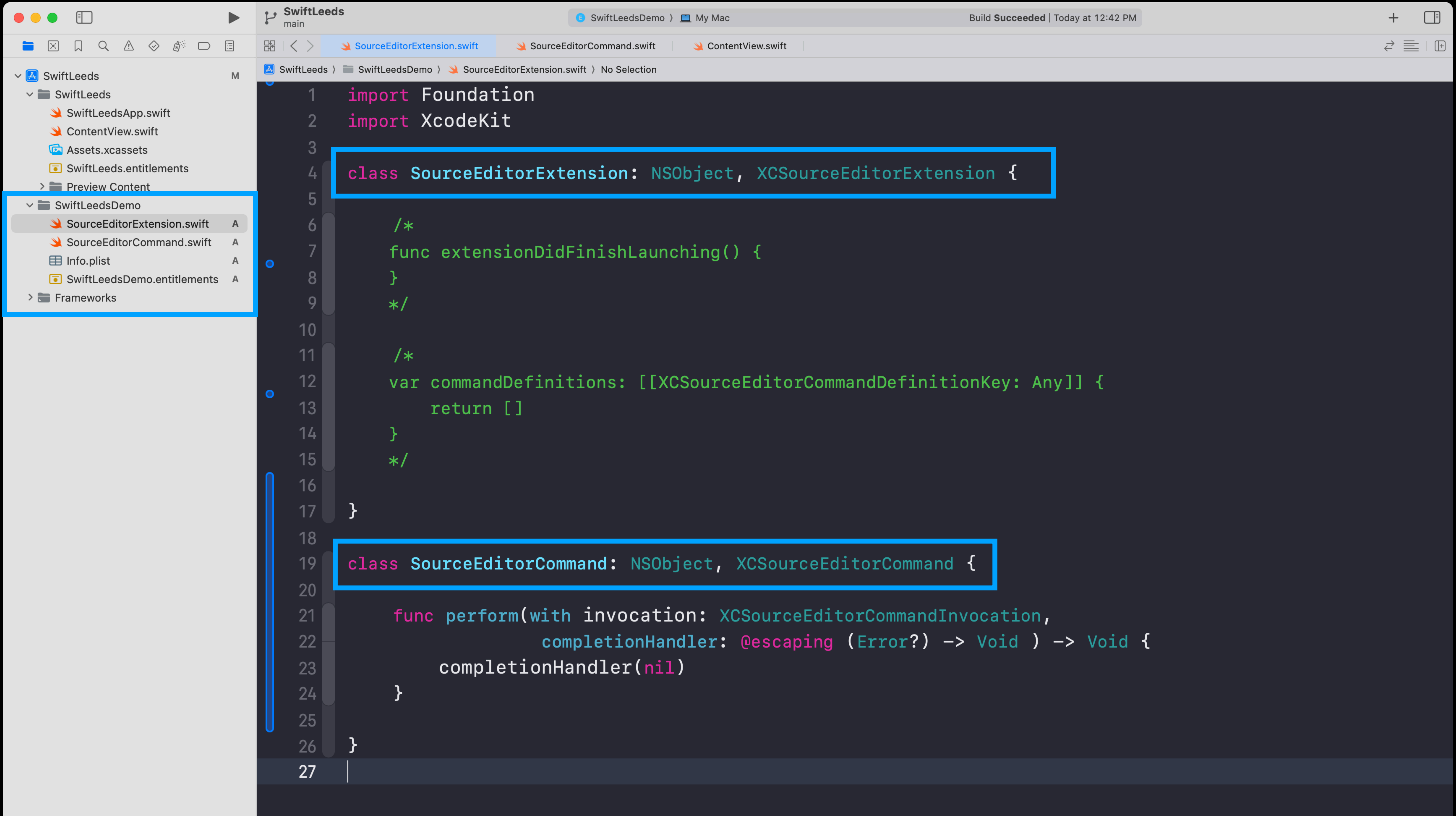
What We’ll Cover

- 📚 Overview
- 🚀 **Building Editor Extensions**
- 💥 Breaking The Rules
- 📦 Distribution
- 📥 Installation


No Selection
Choose a template for your new project:
Multiplatform
iOS
macOS
watchOS
tvOS
DriverKit
Other
Filter
Application
App
Document App
Game
Command Line Tool
Safari Extension App
Framework & Library
Framework
Library
Metal Library
XPC Service
Bundle
Other
Cancel
Previous
Next



Choose a template for your new target:


SwiftLeeds main — SwiftLeedsDemo › My Mac — Build Succeeded | Today at 12:42 PM

SourceEditorExtension.swift | SourceEditorCommand.swift | ContentView.swift

SwiftLeeds › SwiftLeedsDemo › SourceEditorExtension.swift › No Selection

```
SwiftLeeds                                  M
  SwiftLeeds
    SwiftLeedsApp.swift
    ContentView.swift
    Assets.xcassets
    SwiftLeeds.entitlements
    Preview Content
  SwiftLeedsDemo
    SourceEditorExtension.swift             A
    SourceEditorCommand.swift               A
    Info.plist                              A
    SwiftLeedsDemo.entitlements             A
  Frameworks
```

```swift
import Foundation
import XcodeKit

class SourceEditorExtension: NSObject, XCSourceEditorExtension {

    /*
    func extensionDidFinishLaunching() {
    }
    */

    /*
    var commandDefinitions: [[XCSourceEditorCommandDefinitionKey: Any]] {
        return []
    }
    */

}

class SourceEditorCommand: NSObject, XCSourceEditorCommand {

    func perform(with invocation: XCSourceEditorCommandInvocation,
                    completionHandler: @escaping (Error?) -> Void ) -> Void {
        completionHandler(nil)
    }

}
```

🔒 Permissions

Entitlements

⚙️ Configuration

XCSourceEditorExtension

Info.plist

🛠 Implementation

SourceEditorCommand

🔒

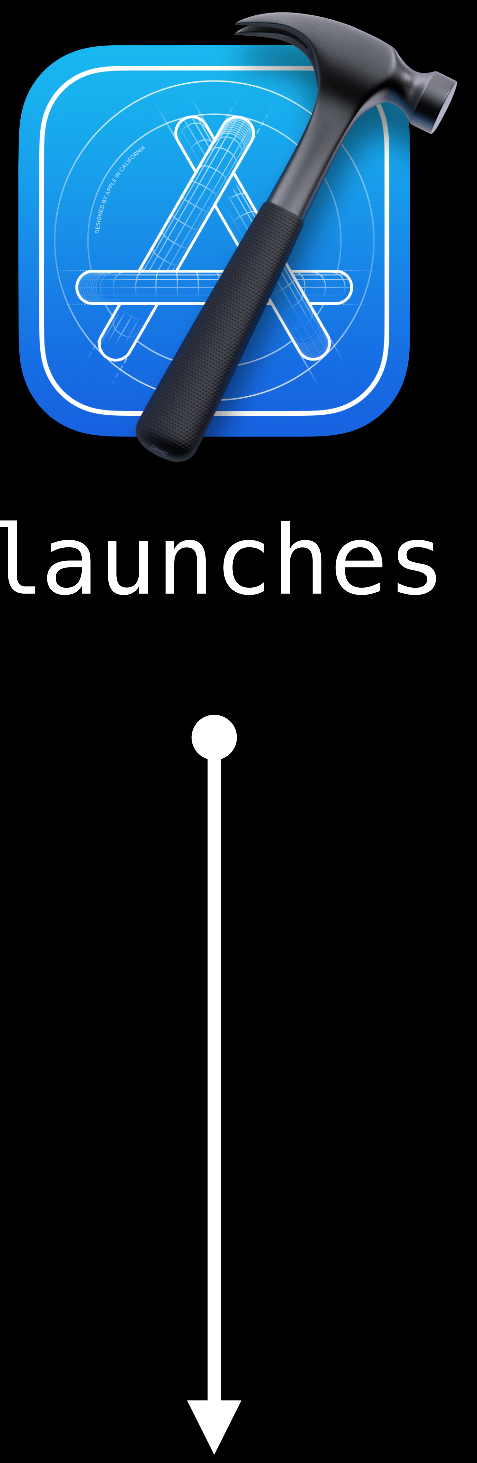
Permissions

```
<plist version="1.0">
<dict>
  <key>com.apple.security.app-sandbox</key>
  <true/>
</dict>
</plist>
```

`.entitlements`

Source Editor Extensions must operate within the App Sandbox 🏝

Configuration


launches

XCSourceEditorExtension

Handles extension launch and specifies the available commands launches

XCSourceEditorExtension

Handles extension launch and specifies the available commands

```swift
protocol XCSourceEditorExtension {
    /// Tells the extension that it successfully launched and may begin to
    /// receive editor commands.
    func extensionDidFinishLaunching()





}
```

```swift
protocol XCSourceEditorExtension {
    /// Tells the extension that it successfully launched and may begin to
    /// receive editor commands.
    func extensionDidFinishLaunching()

    /// This is an array that maps command names to their implementations
    /// in the extension.
    ///
    /// Use this property if you want to customize the available commands at launch time.
    var commandDefinitions: [[XCSourceEditorCommandDefinitionKey: Any]]
}
```

Providing Commands

XCSourceEditorExtension.swift

```swift
class SourceEditorExtension: NSObject, XCSourceEditorExtension {
    var commandDefinitions: [[XCSourceEditorCommandDefinitionKey: Any]] {
        return [
            [
                .classNameKey: "SwiftLeedsDemo.SourceEditorCommand",
                .identifierKey: "com.AryamanSharda.SwiftLeeds.SwiftLeedsDemo",
                .nameKey: "Command #1",
            ]
        ]
    }
}
```

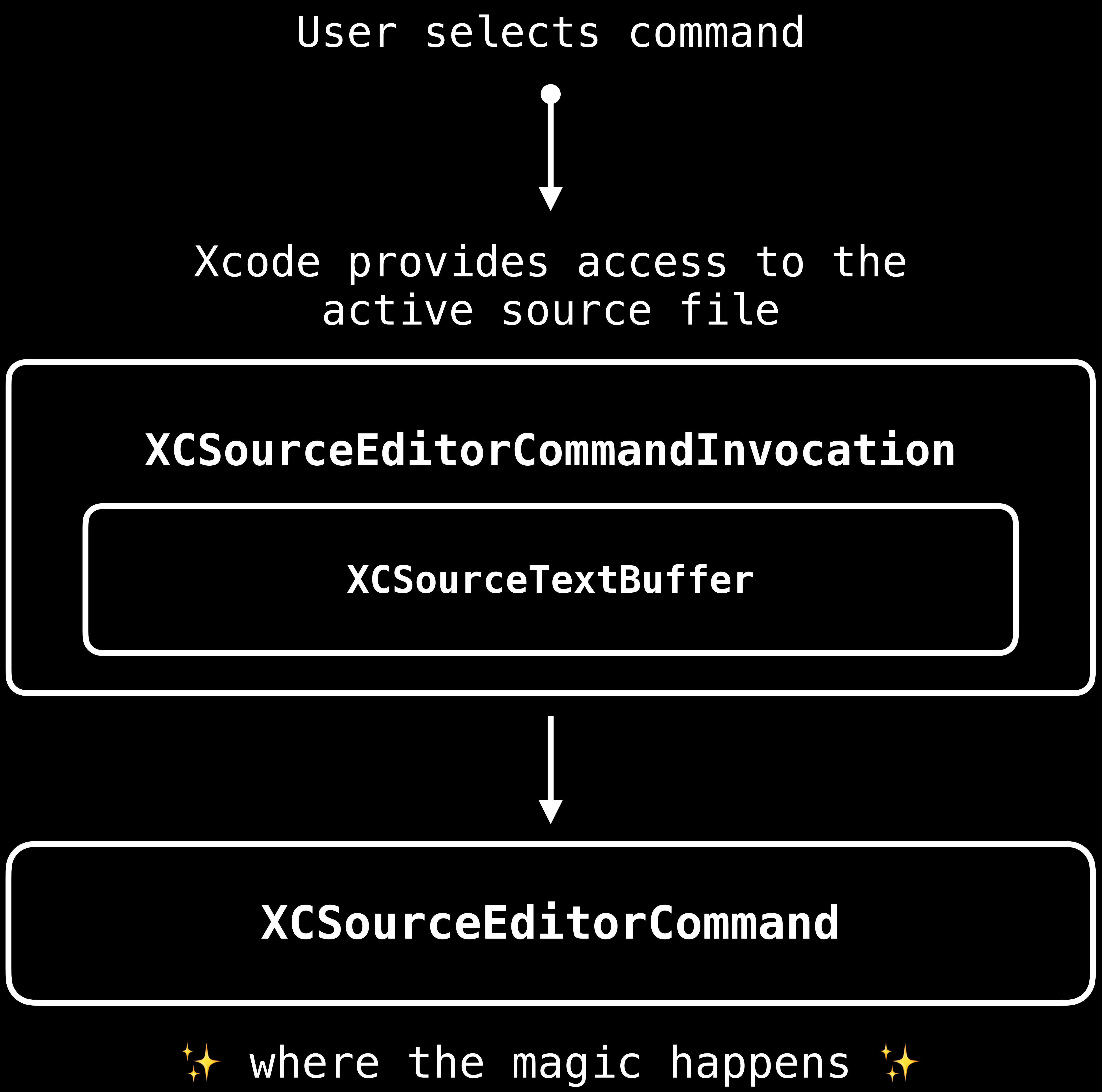
Providing Commands

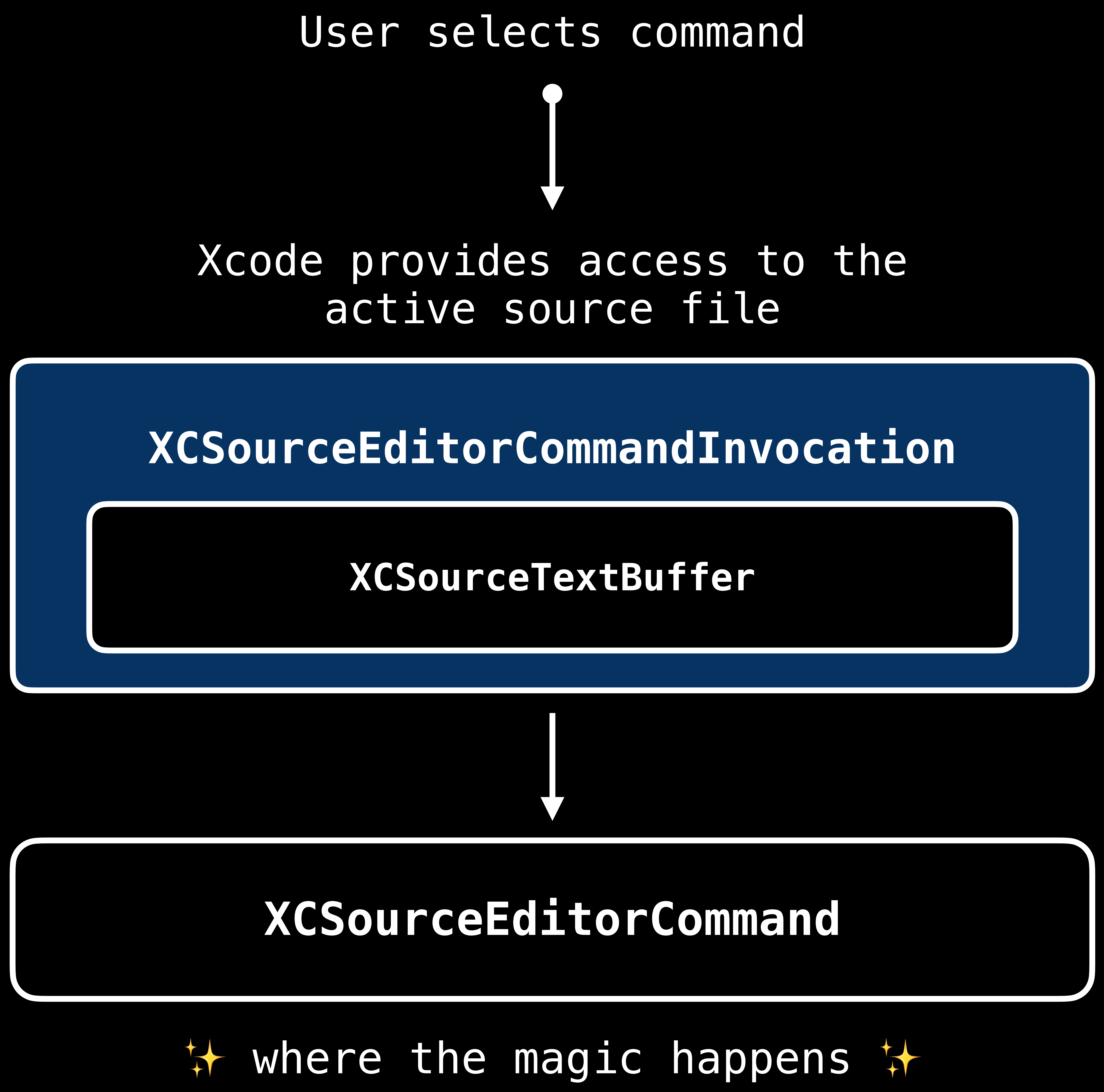
Info.plist

```
<key>NSExtension</key>
   <dict>
      <key>NSExtensionAttributes</key>
      <dict>
         <key>XCSourceEditorCommandDefinitions</key>
         <array>
            <dict>
               <key>XCSourceEditorCommandClassName</key>
               <string>$(PRODUCT_MODULE_NAME).SourceEditorCommand</string>
               <key>XCSourceEditorCommandIdentifier</key>
               <string>$(PRODUCT_BUNDLE_IDENTIFIER).SourceEditorCommand</string>
               <key>XCSourceEditorCommandName</key>
               <string>Source Editor Command</string>
            </dict>
         </array>
         <key>XCSourceEditorExtensionPrincipalClass</key>
         <string>$(PRODUCT_MODULE_NAME).SourceEditorExtension</string>
      </dict>
      <key>NSExtensionPointIdentifier</key>
      <string>com.apple.dt.Xcode.extension.source-editor</string>
   </dict>
```

🛠️

Implementation

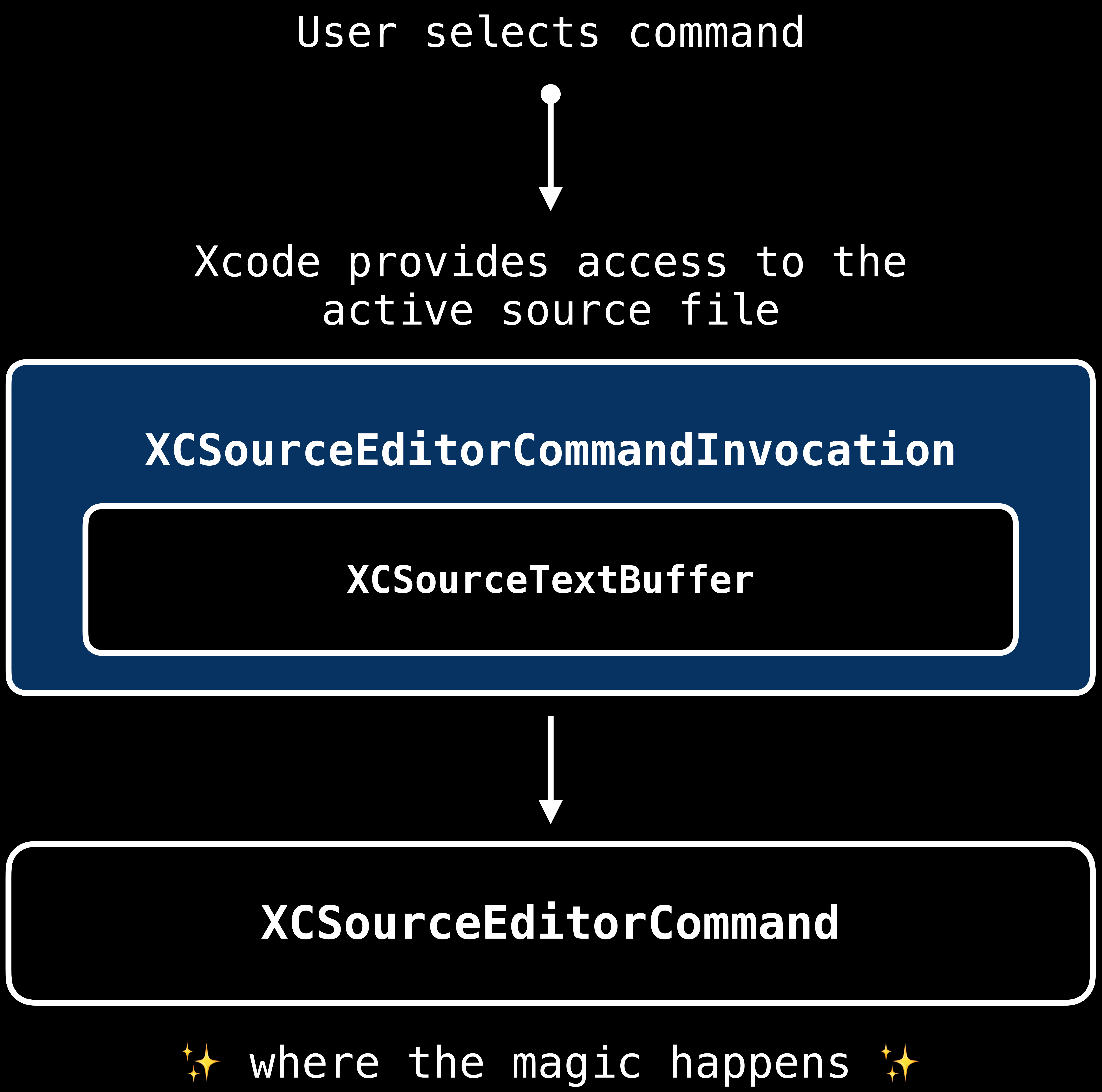

User selects command
Xcode provides access to the active source file
XCSourceEditorCommandInvocation
XCSourceTextBuffer
XCSourceEditorCommand
✨ where the magic happens ✨

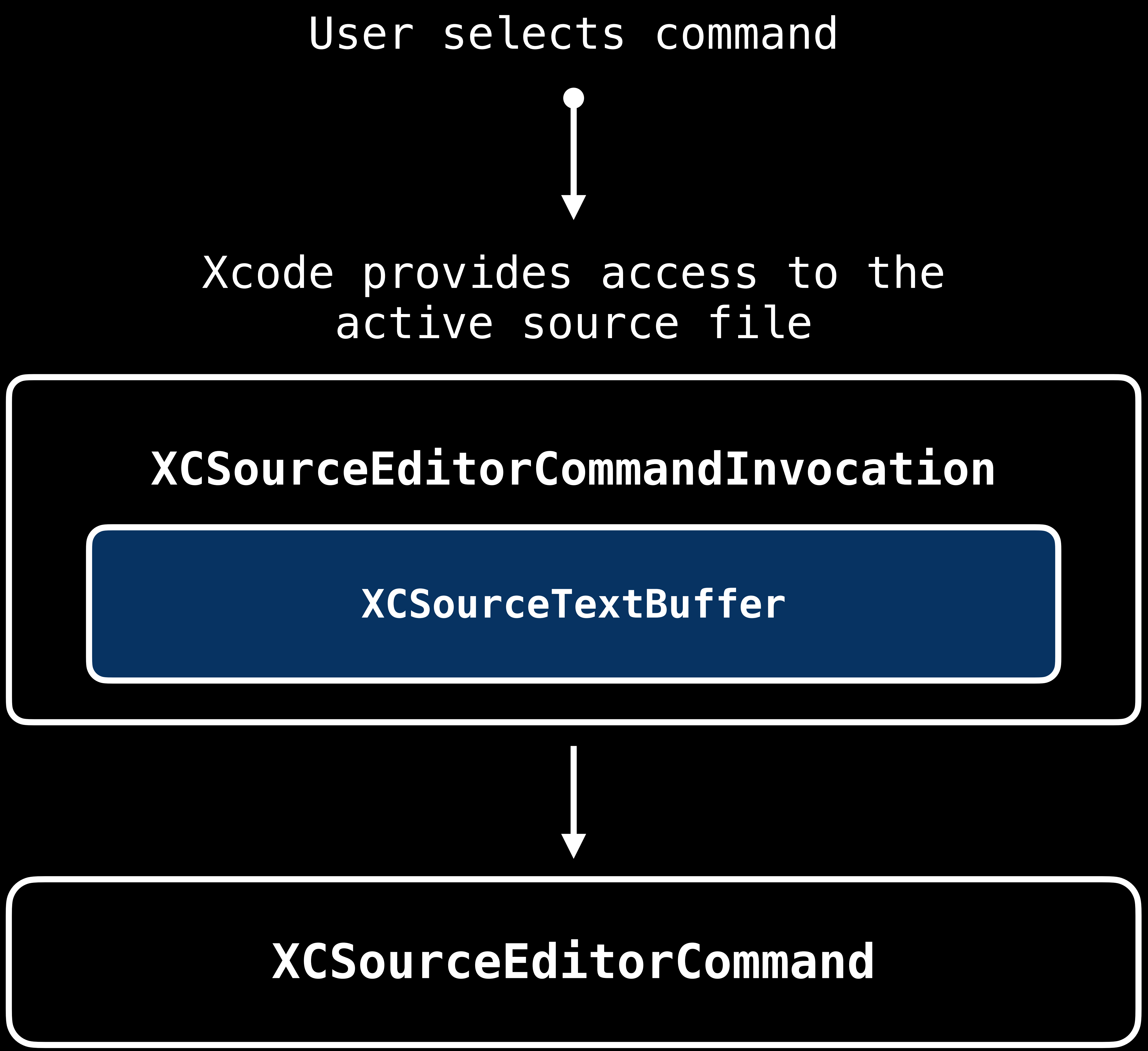

User selects command
Xcode provides access to the active source file
XCSourceEditorCommandInvocation
XCSourceTextBuffer
XCSourceEditorCommand
✨ where the magic happens ✨

```swift
class XCSourceEditorCommandInvocation: NSObject {

    /// The identifier of the command that the user invoked.
    private(set) var commandIdentifier: String?

    /// The buffer of source text upon which the command can operate.
    private(set) var buffer: XCSourceTextBuffer?

}
```

```swift
class XCSourceEditorCommandInvocation: NSObject {

    /// The identifier of the command that the user invoked.
    private(set) var commandIdentifier: String?

    /// The buffer of source text upon which the command can operate.
    private(set) var buffer: XCSourceTextBuffer?

}
```

```swift
class XCSourceEditorCommandInvocation: NSObject {

    /// The identifier of the command that the user invoked.
    private(set) var commandIdentifier: String?

    /// The buffer of source text upon which the command can operate.
    private(set) var buffer: XCSourceTextBuffer?

}
```

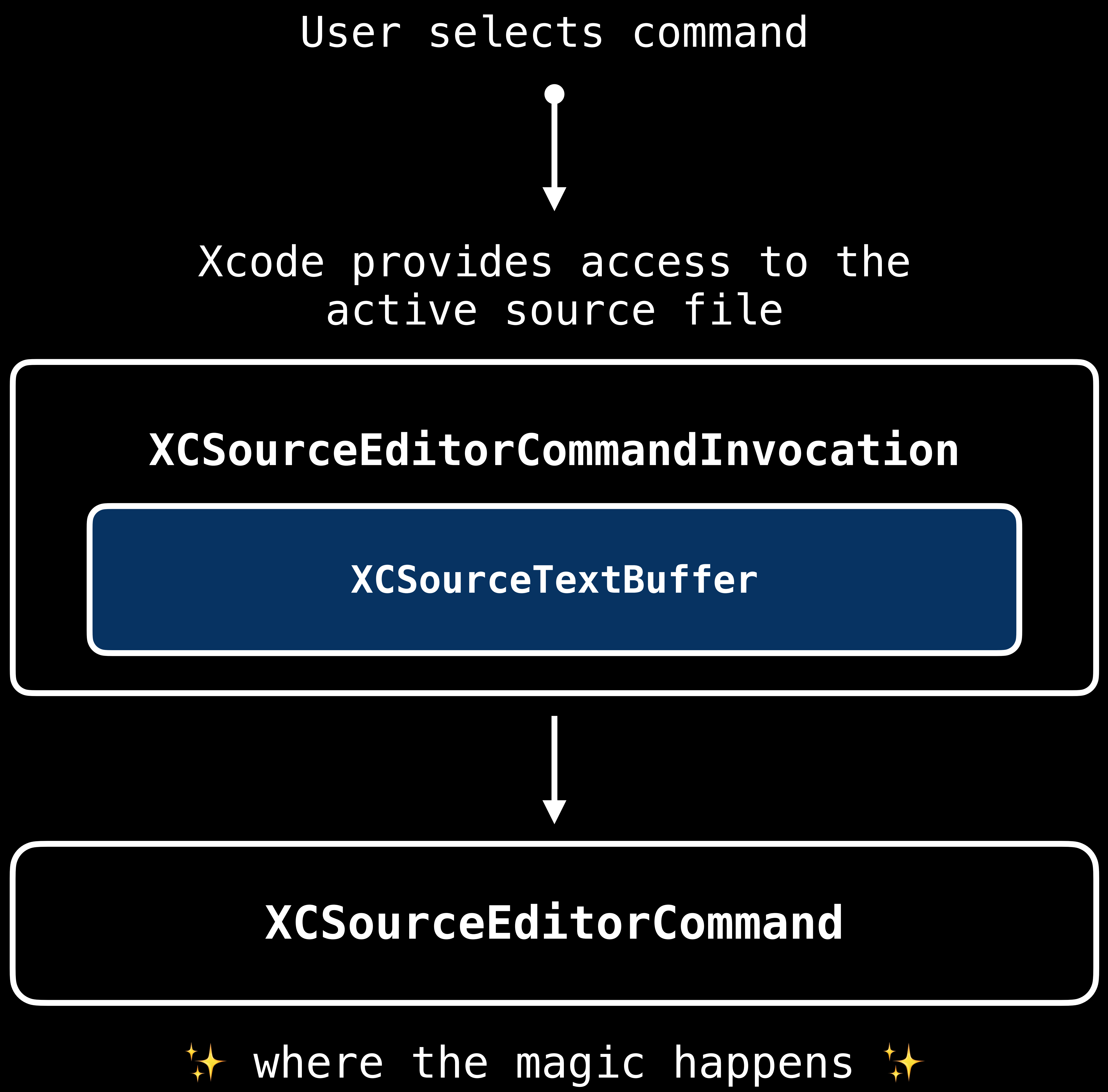

User selects command
Xcode provides access to the active source file
XCSourceEditorCommandInvocation
XCSourceTextBuffer
XCSourceEditorCommand
✨ where the magic happens ✨

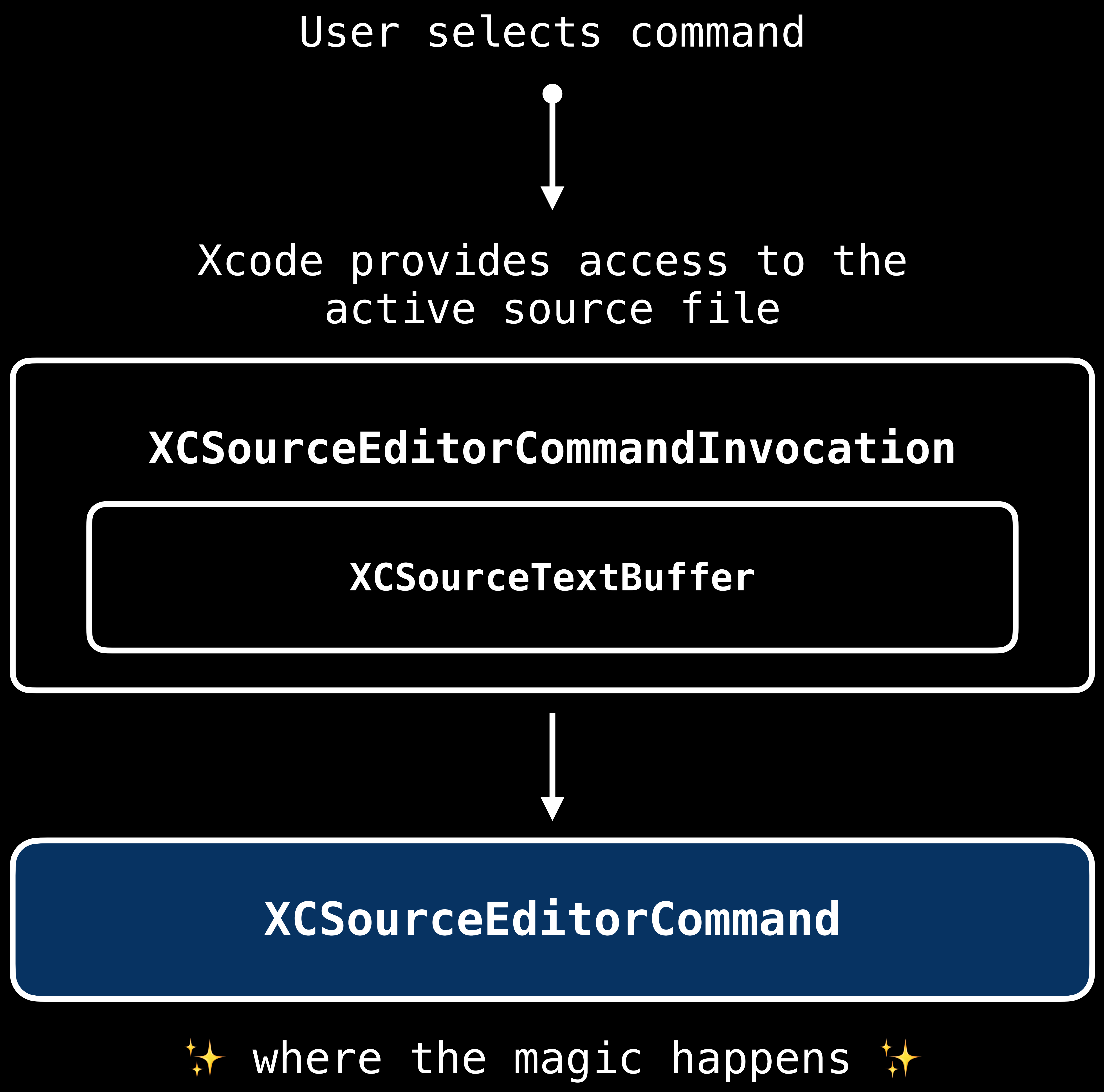

User selects command
Xcode provides access to the active source file
XCSourceEditorCommandInvocation
XCSourceTextBuffer
XCSourceEditorCommand

✨ where the magic happens ✨

```swift
class XCSourceTextBuffer: NSObject {

    /// Spaces per tab
    let tabWidth: Int

    /// Spaces for indentation
    let indentationWidth: Int

    /// Use tabs for indentation
    let usesTabsForIndentation: Bool

    ///...
}
```

```swift
class XCSourceTextBuffer: NSObject {
    /// The type or format of the content stored in the buffer
    /// ex. plain text, Swift code, HTML, etc.
    let contentUTI: String

    /// The lines of text in the buffer, including line endings.
    var lines: [String]

    /// The text selections in the buffer.
    var selections: [XCSourceTextRange]

    /// The complete buffer's string representation.
    ///
    /// Changes to the `lines` property are immediately reflected in this property
    /// and vice versa.
    var completeBuffer: String
}
```


User selects command
Xcode provides access to the active source file
XCSourceEditorCommandInvocation
XCSourceTextBuffer
XCSourceEditorCommand
✨ where the magic happens ✨

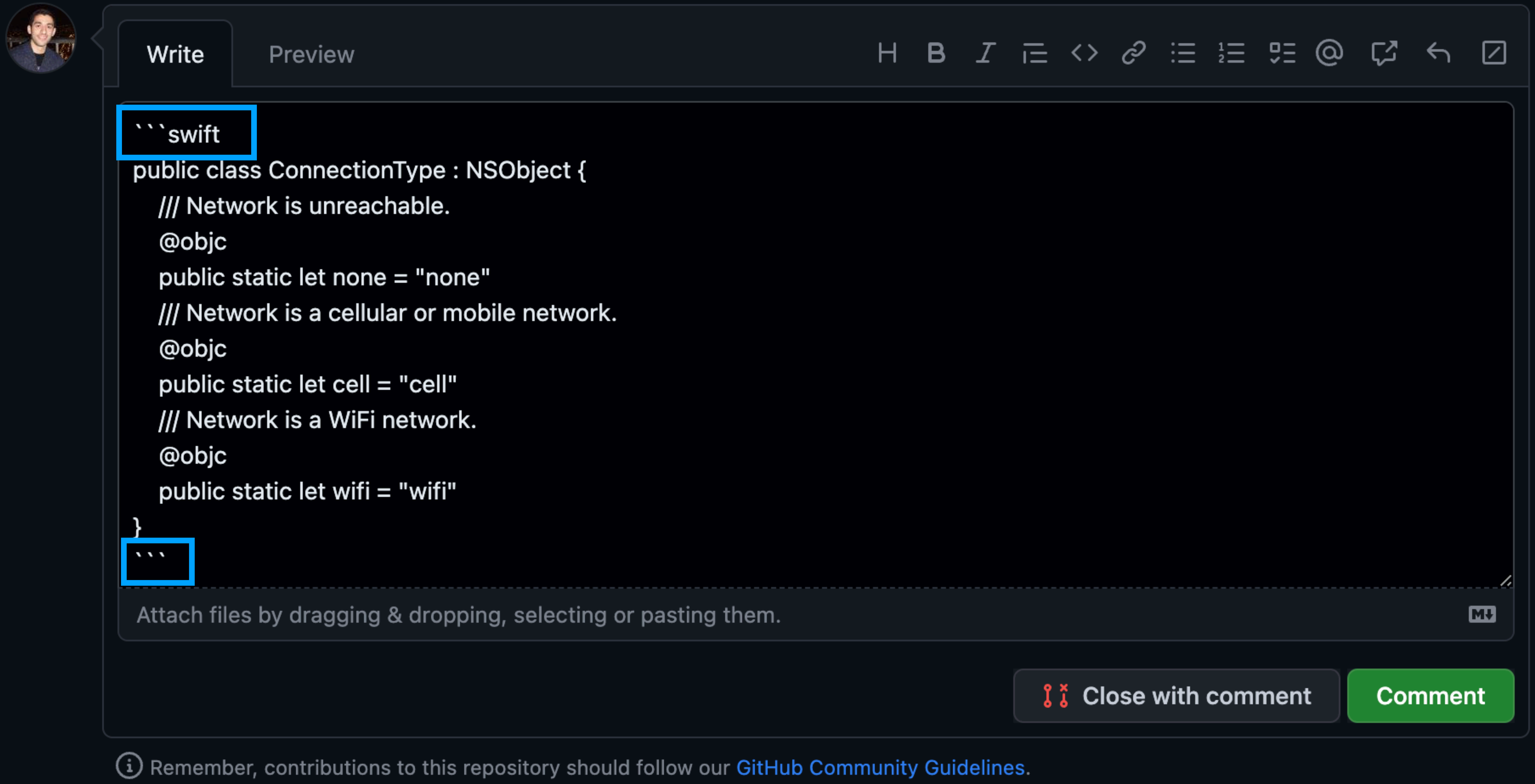

User selects command
Xcode provides access to the active source file
XCSourceEditorCommandInvocation
XCSourceTextBuffer
XCSourceEditorCommand
✨ where the magic happens ✨

```swift
protocol XCSourceEditorCommand {

    /// Performs the action associated with the command
    /// using the information in the invocation.
    func perform(
        with: XCSourceEditorCommandInvocation,
        completionHandler: (Error?) -> Void
    )
}
```

```swift
protocol XCSourceEditorCommand {

    /// Performs the action associated with the command
    /// using the information in the CommandInvocation.
    func perform(
        with: XCSourceEditorCommandInvocation,
        completionHandler: (Error?) -> Void
    )
}
```

🚀

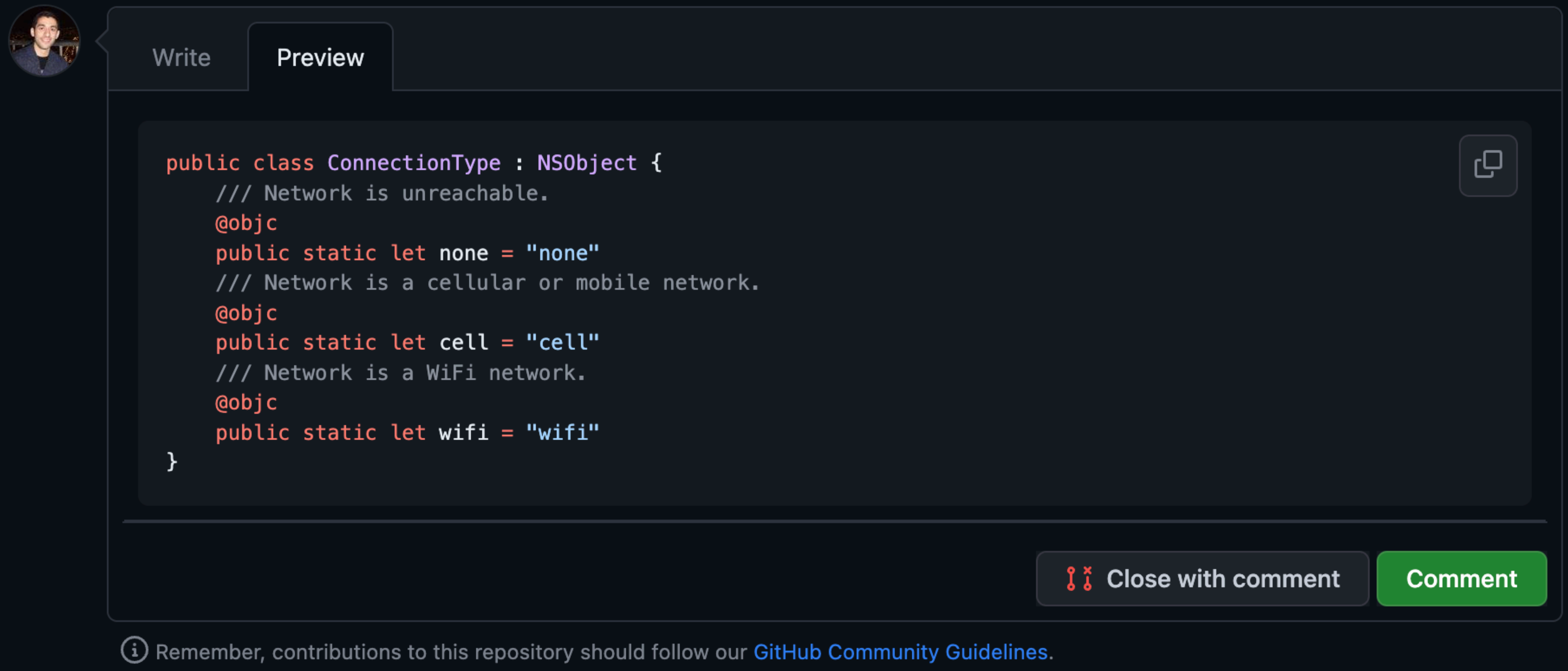
Auto-Formatting Code Snippets



~~~
```swift
public class ConnectionType : NSObject {
    /// Network is unreachable.
    @objc
    public static let none = "none"
    /// Network is a cellular or mobile network.
    @objc
    public static let cell = "cell"
    /// Network is a WiFi network.
    @objc
    public static let wifi = "wifi"
}
```
~~~

Attach files by dragging & dropping, selecting or pasting them.

Remember, contributions to this repository should follow our GitHub Community Guidelines.



```swift
public class ConnectionType : NSObject {
    /// Network is unreachable.
    @objc
    public static let none = "none"
    /// Network is a cellular or mobile network.
    @objc
    public static let cell = "cell"
    /// Network is a WiFi network.
    @objc
    public static let wifi = "wifi"
}
```

Remember, contributions to this repository should follow our GitHub Community Guidelines.

```swift
func perform(with invocation: XCSourceEditorCommandInvocation, completionHandler: (Error?) -> Void) {

    guard let selections = invocation.buffer.selections as? [XCSourceTextRange],
          let selection = selections.first else {
        completionHandler(CopyAsMarkdownError.noSelection.nsError)
        return
    }

    let startIndex = selection.start.line
    let endIndex = selection.end.line
    let selectedRange = NSRange(location: startIndex, length: 1 + endIndex - startIndex)

    // Grabs the lines included in the current selection
    guard let selectedLines = invocation.buffer.lines.subarray(with: selectedRange) as? [String] else {
        completionHandler(CopyAsMarkdownError.failedToCastSelection.nsError)
        return
    }

    // Adds the Markdown formatting and assigns it to the clipboard
    let text = selectedLines.joined()
    let pasteboardString = "```\n\(text)```"
    let pasteboard = NSPasteboard.general
    pasteboard.declareTypes([.string], owner: nil)
    pasteboard.setString(pasteboardString, forType: .string)

    completionHandler(nil)
}
```

```swift
func perform(with invocation: XCSourceEditorCommandInvocation, completionHandler: (Error?) -> Void) {
    guard let selections = invocation.buffer.selections as? [XCSourceTextRange],
          let selection = selections.first else {
        completionHandler(CopyAsMarkdownError.noSelection.nsError)
        return
    }

    let startIndex = selection.start.line
    let endIndex = selection.end.line
    let selectedRange = NSRange(location: startIndex, length: 1 + endIndex - startIndex)

    // Grabs the lines included in the current selection
    guard let selectedLines = invocation.buffer.lines.subarray(with: selectedRange) as? [String] else {
        completionHandler(CopyAsMarkdownError.failedToCastSelection.nsError)
        return
    }

    // Adds the Markdown formatting and assigns it to the clipboard
    let text = selectedLines.joined()
    let pasteboardString = "```\n\(text)```"
    let pasteboard = NSPasteboard.general
    pasteboard.declareTypes([.string], owner: nil)
    pasteboard.setString(pasteboardString, forType: .string)

    completionHandler(nil)
}
```

```swift
func perform(with invocation: XCSourceEditorCommandInvocation, completionHandler: (Error?) -> Void) {

    guard let selections = invocation.buffer.selections as? [XCSourceTextRange],
          let selection = selections.first else {
        completionHandler(CopyAsMarkdownError.noSelection.nsError)
        return
    }

    let startIndex = selection.start.line
    let endIndex = selection.end.line
    let selectedRange = NSRange(location: startIndex, length: 1 + endIndex - startIndex)

    // Grabs the lines included in the current selection
    guard let selectedLines = invocation.buffer.lines.subarray(with: selectedRange) as? [String] else {
        completionHandler(CopyAsMarkdownError.failedToCastSelection.nsError)
        return
    }

    // Adds the Markdown formatting and assigns it to the clipboard
    let text = selectedLines.joined()
    let pasteboardString = "```\n\(text)```"
    let pasteboard = NSPasteboard.general
    pasteboard.declareTypes([.string], owner: nil)
    pasteboard.setString(pasteboardString, forType: .string)

    completionHandler(nil)
}
```

```swift
func perform(with invocation: XCSourceEditorCommandInvocation, completionHandler: (Error?) -> Void) {

    guard let selections = invocation.buffer.selections as? [XCSourceTextRange],
          let selection = selections.first else {
        completionHandler(CopyAsMarkdownError.noSelection.nsError)
        return
    }

    let startIndex = selection.start.line
    let endIndex = selection.end.line
    let selectedRange = NSRange(location: startIndex, length: 1 + endIndex - startIndex)

    // Grabs the lines included in the current selection
    guard let selectedLines = invocation.buffer.lines.subarray(with: selectedRange) as? [String] else {
        completionHandler(CopyAsMarkdownError.failedToCastSelection.nsError)
        return
    }

    // Adds the Markdown formatting and assigns it to the clipboard
    let text = selectedLines.joined()
    let pasteboardString = "```\n\(text)```"
    let pasteboard = NSPasteboard.general
    pasteboard.declareTypes([.string], owner: nil)
    pasteboard.setString(pasteboardString, forType: .string)

    completionHandler(nil)
}
```

```swift
func perform(with invocation: XCSourceEditorCommandInvocation, completionHandler: (Error?) -> Void) {

    guard let selections = invocation.buffer.selections as? [XCSourceTextRange],
          let selection = selections.first else {
        completionHandler(CopyAsMarkdownError.noSelection.nsError)
        return
    }

    let startIndex = selection.start.line
    let endIndex = selection.end.line
    let selectedRange = NSRange(location: startIndex, length: 1 + endIndex - startIndex)

    // Grabs the lines included in the current selection
    guard let selectedLines = invocation.buffer.lines.subarray(with: selectedRange) as? [String] else {
        completionHandler(CopyAsMarkdownError.failedToCastSelection.nsError)
        return
    }

    // Adds the Markdown formatting and assigns it to the clipboard
    let text = selectedLines.joined()
    let pasteboardString = "```\n\(text)```"
    let pasteboard = NSPasteboard.general
    pasteboard.declareTypes([.string], owner: nil)
    pasteboard.setString(pasteboardString, forType: .string)

    completionHandler(nil)
}
```

```swift
func perform(with invocation: XCSourceEditorCommandInvocation, completionHandler: (Error?) -> Void) {

    guard let selections = invocation.buffer.selections as? [XCSourceTextRange],
          let selection = selections.first else {
        completionHandler(CopyAsMarkdownError.noSelection.nsError)
        return
    }

    let startIndex = selection.start.line
    let endIndex = selection.end.line
    let selectedRange = NSRange(location: startIndex, length: 1 + endIndex - startIndex)

    // Grabs the lines included in the current selection
    guard let selectedLines = invocation.buffer.lines.subarray(with: selectedRange) as? [String] else {
        completionHandler(CopyAsMarkdownError.failedToCastSelection.nsError)
        return
    }

    // Adds the Markdown formatting and assigns it to the clipboard
    let text = selectedLines.joined()
    let pasteboardString = "```\n\(text)```"
    let pasteboard = NSPasteboard.general
    pasteboard.declareTypes([.string], owner: nil)
    pasteboard.setString(pasteboardString, forType: .string)

    completionHandler(nil)
}
```

Xcode
File
Edit
View
Find
Navigate
Editor
Product
Debug
Integrate
Window
Help
Sat Sep 30 8:24 PM
SwiftLeeds
main
SwiftLeedsDemo
My Mac
Build Succeeded | 9/27/23 at 12:42 PM
SourceEditorExtension.swift
Info.plist
SourceEditorCommand.swift
ContentView.swift
SwiftLeedsApp.swift
SwiftLeeds
SwiftLeeds
SwiftLeedsApp.swift
SwiftLeedsApp
SwiftLeeds M
SwiftLeeds
SwiftLeedsApp.swift
ContentView.swift M
Assets.xcassets
SwiftLeeds.entitlements
Preview Content
SwiftLeedsDemo
SourceEditorExtension.swift A
SourceEditorCommand.swift A
Info.plist A
SwiftLeedsDemo.entitlements A
Frameworks
Filter
Inferior
Line: 11 Col: 1

```swift
//
//  SwiftLeedsApp.swift
//  SwiftLeeds
//
//  Created by Aryaman Sharda on 9/27/23.
//

import SwiftUI

@main
struct SwiftLeedsApp: App {
    var body: some Scene {
        WindowGroup {
            ContentView()
        }
    }
}
```

Testing

EditKit 〉 My Mac   Finished running EditKit ⚠ 3

FormatCodeForSharingCommand.swift   EditKitPro.xcodeproj

General   Signing & Capabilities   Resource Tags   Info   Build Settings   Build Phases   Build Rules

ons

Destination	SDK
Mac	macOS

ts

macOS 12.6

aries

Name	Embed
Cocoa.framework	Do Not Embed
XcodeKit.framework	Embed & Sign

Do Not Embed
✓ Embed & Sign
Embed Without Signing

Add development assets here

EditKit
My Mac
FormatCodeForSharingCommand.swift
ommand.swift 〉 No Selection
Filter
EditKitPro
✓ EditKit
Edit Scheme...
New Scheme...
Manage Schemes...

```
ng.swift


 Sharda on 12/17/22.






eForSharingCommand {
rm(with invocation: XCSourceEditorCommandInvocation, completionHandler: (Error?) -> Void) {
lections = invocation.buffer.selections as? [XCSourceTextRange], let selection = selections.first else
ionHandler(GenericError.default.nsError)



lex = selection.start.line
 = selection.end.line
Range = NSRange(location: startIndex, length: 1 + endIndex - startIndex)
```

```
d {
ion: XCSourceEditorCommandInvocation, completionHandler: (Error?) -> Void) {
cation.buffer.selections as? [XCSourceTextRange], let selection = selections.first else {
icError.default.nsError)



.start.line
d.line
(location: startIndex, length: 1 + endIndex - star

n the current selection
nvocation.buffer.lines.subarray(with: selectedRange
icError.default.nsError)



 String with the formatting stripped away
ined()
\n\(text)```"
rd.general
ring], owner: nil)
oardString, forType: .string)
```

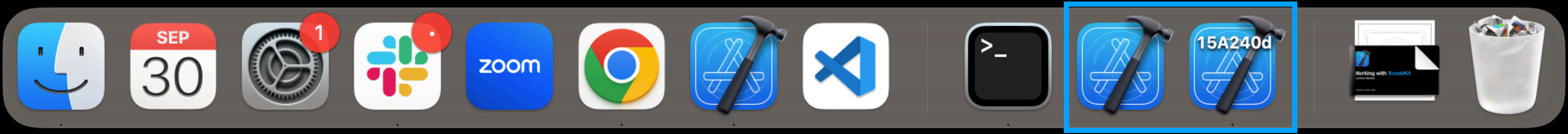
Choose an app to run:

Recent Applications
- Xcode
- Safari

All Applications
- 1Password
- Acrobat Reader
- App Store
- Arduino
- ASO Toolkit
- Astro
- Audacity

Filter

Browse... | Cancel | Run


SEP
30
1
zoom
15A240d
Working with XcodeKit

```
        if let firstLine = lines.first {
            let leadingWhitespaceRegex = try! NSRegularExpression(pattern: "^[ \\t]+", options: .anchorsMatchLines)
            if let match = leadingWhitespaceRegex.firstMatch(in: firstLine, options: [], range: NSRange(firstLine.startIndex..<firstL
                commonLeadingWhitespace = String(firstLine[Range(match.range, in: firstLine)!])
            }
        }

        // Remove the com
        let transformedL
            guard line.h
                return l
            }
            return Strin
        }

        // Join the lines
        let transformedT

        return transform
    }
}
```

Build Configuration: Debug

Executable: Xcode.app

☐ Debug executable

Debug Process As: ● Me (aryamansharda) ○ root

LLDB Init File: $(SRCROOT)/LLDBInitFile

Launch: [✓ Automatically] ○ Wait for the executable to be launched

```
Bootstrapping; external subsystem UIKit_PKSubsystem refused setup
```

```
        var commonLeadingWhitespace = ""
        if let firstLine = lines.first {
            let leadingWhitespaceRegex = try! NSRegularExpression(pattern: "^[ \\t]+", options: .anchorsMatchLines)
            if let match = leadingWhitespaceRegex.firstMatch(in: firstLine, options: [], range: NSRange(firstLine.startIndex..<firstLi
                commonLeadingWhitespace = String(firstLine[Range(match.range, in: firstLine)!])
            }
        }

        // Remove the com
        let transformedL
            guard line.
                return l
            }
            return Strin
        }

        // Join the lines
        let transformedT

        return transform
    }
}
```

Arguments Passed On Launch

☑ /Users/aryamansharda/Documents/Personal/BuildSwitcher/BuildSwitcher.xcodeproj

Environment Variables

Name	Value
No Environment Variables	

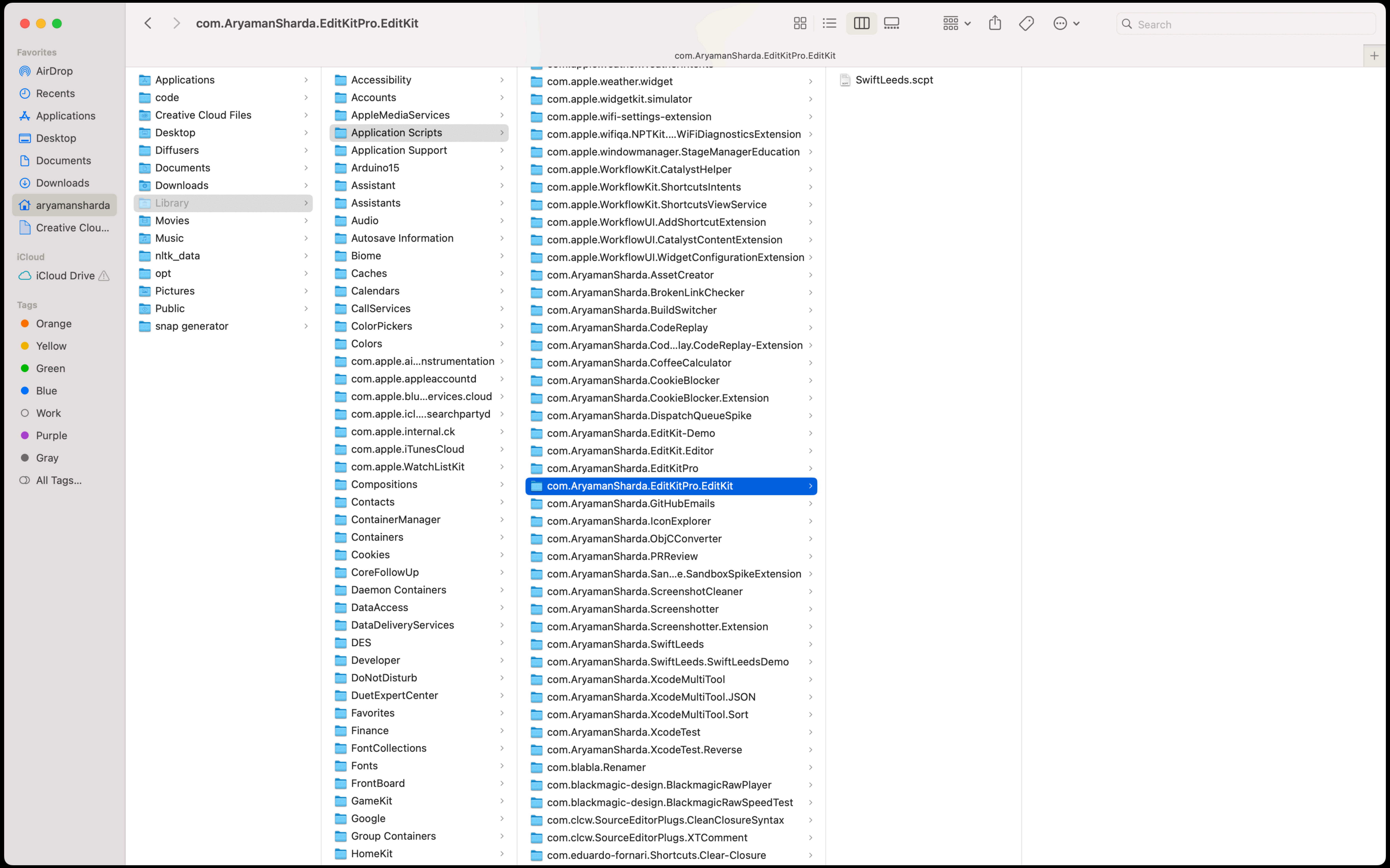
Expand Variables Based On: EditKitPro

```
Bootstrapping; external subsystem UIKit_PKSubsystem refused setup
Program ended with exit code: -1
```

What We'll Cover

- 📚 Overview
- 🚀 Building Editor Extensions
- 💥 **Breaking The Rules**
- 📦 Distribution
- 📥 Installation

AppleScript



Applications
code
Creative Cloud Files
Desktop
Diffusers
Documents
Downloads
Library
Movies
Music
nltk_data
opt
Pictures
Public
snap generator

Accessibility
Accounts
AppleMediaServices
Application Scripts
Application Support
Arduino15
Assistant
Assistants
Audio
Autosave Information
Biome
Caches
Calendars
CallServices
ColorPickers
Colors
com.apple.ai...nstrumentation
com.apple.appleaccountd
com.apple.blu...ervices.cloud
com.apple.icl....searchpartyd
com.apple.internal.ck
com.apple.iTunesCloud
com.apple.WatchListKit
Compositions
Contacts
ContainerManager
Containers
Cookies
CoreFollowUp
Daemon Containers
DataAccess
DataDeliveryServices
DES
Developer
DoNotDisturb
DuetExpertCenter
Favorites
Finance
FontCollections
Fonts
FrontBoard
GameKit
Google
Group Containers
HomeKit

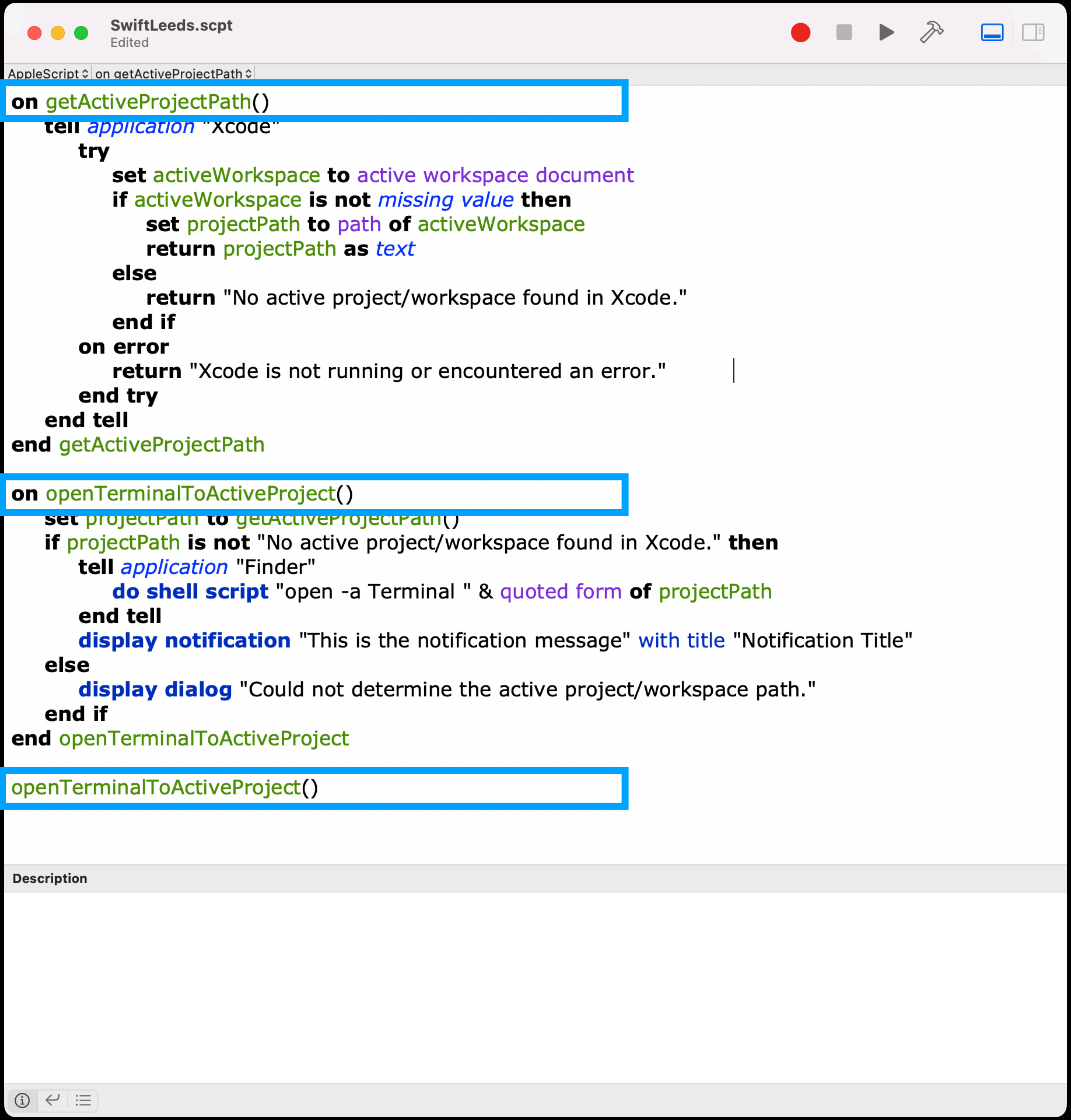
com.apple.weather.widget
com.apple.widgetkit.simulator
com.apple.wifi-settings-extension
com.apple.wifiqa.NPTKit....WiFiDiagnosticsExtension
com.apple.windowmanager.StageManagerEducation
com.apple.WorkflowKit.CatalystHelper
com.apple.WorkflowKit.ShortcutsIntents
com.apple.WorkflowKit.ShortcutsViewService
com.apple.WorkflowUI.AddShortcutExtension
com.apple.WorkflowUI.CatalystContentExtension
com.apple.WorkflowUI.WidgetConfigurationExtension
com.AryamanSharda.AssetCreator
com.AryamanSharda.BrokenLinkChecker
com.AryamanSharda.BuildSwitcher
com.AryamanSharda.CodeReplay
com.AryamanSharda.Cod...lay.CodeReplay-Extension
com.AryamanSharda.CoffeeCalculator
com.AryamanSharda.CookieBlocker
com.AryamanSharda.CookieBlocker.Extension
com.AryamanSharda.DispatchQueueSpike
com.AryamanSharda.EditKit-Demo
com.AryamanSharda.EditKit.Editor
com.AryamanSharda.EditKitPro
com.AryamanSharda.EditKitPro.EditKit
com.AryamanSharda.GitHubEmails
com.AryamanSharda.IconExplorer
com.AryamanSharda.ObjCConverter
com.AryamanSharda.PRReview
com.AryamanSharda.San...e.SandboxSpikeExtension
com.AryamanSharda.ScreenshotCleaner
com.AryamanSharda.Screenshotter
com.AryamanSharda.Screenshotter.Extension
com.AryamanSharda.SwiftLeeds
com.AryamanSharda.SwiftLeeds.SwiftLeedsDemo
com.AryamanSharda.XcodeMultiTool
com.AryamanSharda.XcodeMultiTool.JSON
com.AryamanSharda.XcodeMultiTool.Sort
com.AryamanSharda.XcodeTest
com.AryamanSharda.XcodeTest.Reverse
com.blabla.Renamer
com.blackmagic-design.BlackmagicRawPlayer
com.blackmagic-design.BlackmagicRawSpeedTest
com.clcw.SourceEditorPlugs.CleanClosureSyntax
com.clcw.SourceEditorPlugs.XTComment
com.eduardo-fornari.Shortcuts.Clear-Closure

SwiftLeeds.scpt

```
/Users/aryamansharda/Library/Application Scripts/
com.AryamanSharda.EditKitPro.EditKit
```

Xcode
File
Edit
View
Find
Navigate
Editor
Product
Debug
Integrate
Window
Help
Mon Oct 9 6:42 AM
SwiftLeeds
main
SwiftLeedsDemo
My Mac
Build Succeeded | 9/27/23 at 8:42 PM
SourceEditorExtension.swift
XCSourceEditorExtension.h
SourceEditorCommand.swift
ContentView.swift
SwiftLeedsApp.swift
SwiftLeeds
SwiftLeeds
SwiftLeedsApp.swift
SwiftLeedsApp
SwiftLeeds M
SwiftLeeds
SwiftLeedsApp.swift
ContentView.swift M
CustomerLocation.swift A
Assets.xcassets
SwiftLeeds.entitlements
Preview Content
SwiftLeedsDemo
Frameworks
Filter
Inferior
Line: 11 Col: 28

```swift
//
//  SwiftLeedsApp.swift
//  SwiftLeeds
//
//  Created by Aryaman Sharda on 9/27/23.
//

import SwiftUI

@main
struct SwiftLeedsApp: App {
    var body: some Scene {
        WindowGroup {
            ContentView()
        }
    }
}
```



```applescript
on getActiveProjectPath()
    tell application "Xcode"
        try
            set activeWorkspace to active workspace document
            if activeWorkspace is not missing value then
                set projectPath to path of activeWorkspace
                return projectPath as text
            else
                return "No active project/workspace found in Xcode."
            end if
        on error
            return "Xcode is not running or encountered an error."
        end try
    end tell
end getActiveProjectPath

on openTerminalToActiveProject()
    set projectPath to getActiveProjectPath()
    if projectPath is not "No active project/workspace found in Xcode." then
        tell application "Finder"
            do shell script "open -a Terminal " & quoted form of projectPath
        end tell
        display notification "This is the notification message" with title "Notification Title"
    else
        display dialog "Could not determine the active project/workspace path."
    end if
end openTerminalToActiveProject

openTerminalToActiveProject()
```

```swift
class AppleScriptRunner {
    static func run() {
        // Get the URL for the AppleScript file
        guard let applicationDirectoryPath = try
                   FileManager.default.url(for: .applicationScriptsDirectory,
                   in: .userDomainMask, appropriateFor: nil, create: true) else {
            return
        }

        let scriptPath = applicationDirectoryPath.appendingPathComponent("SwiftLeeds.scpt")

        // Check if the AppleScript file exists and create an NSUserAppleScriptTask
        guard FileManager.default.fileExists(atPath: scriptPath.path),
                let script = try? NSUserAppleScriptTask(url: scriptPath) else {
            return
        }

        // Execute the AppleScript with a nil event (default subroutine)
        script.execute(withAppleEvent: nil) {_, error in
            if let error = error {
                print(error)
            }
        }
    }
}
```

```swift
class AppleScriptRunner {
    static func run() {
        // Get the URL for the AppleScript file
        guard let applicationDirectoryPath = try
                    FileManager.default.url(for: .applicationScriptsDirectory,
                    in: .userDomainMask, appropriateFor: nil, create: true) else {
            return
        }

        let scriptPath = applicationDirectoryPath.appendingPathComponent("SwiftLeeds.scpt")

        // Check if the AppleScript file exists and create an NSUserAppleScriptTask
        guard FileManager.default.fileExists(atPath: scriptPath.path),
                let script = try? NSUserAppleScriptTask(url: scriptPath) else {
            return
        }

        // Execute the AppleScript with a nil event (default subroutine)
        script.execute(withAppleEvent: nil) {_, error in
            if let error = error {
                print(error)
            }
        }
    }
}
```

```swift
class AppleScriptRunner {
    static func run() {
        // Get the URL for the AppleScript file
        guard let applicationDirectoryPath = try
                  FileManager.default.url(for: .applicationScriptsDirectory,
                  in: .userDomainMask, appropriateFor: nil, create: true) else {
            return
        }

        let scriptPath = applicationDirectoryPath.appendingPathComponent("SwiftLeeds.scpt")

        // Check if the AppleScript file exists and create an NSUserAppleScriptTask
        guard FileManager.default.fileExists(atPath: scriptPath.path),
                let script = try? NSUserAppleScriptTask(url: scriptPath) else {
            return
        }

        // Execute the AppleScript with a nil event (default subroutine)
        script.execute(withAppleEvent: nil) {_, error in
            if let error = error {
                print(error)
            }
        }
    }
}
```

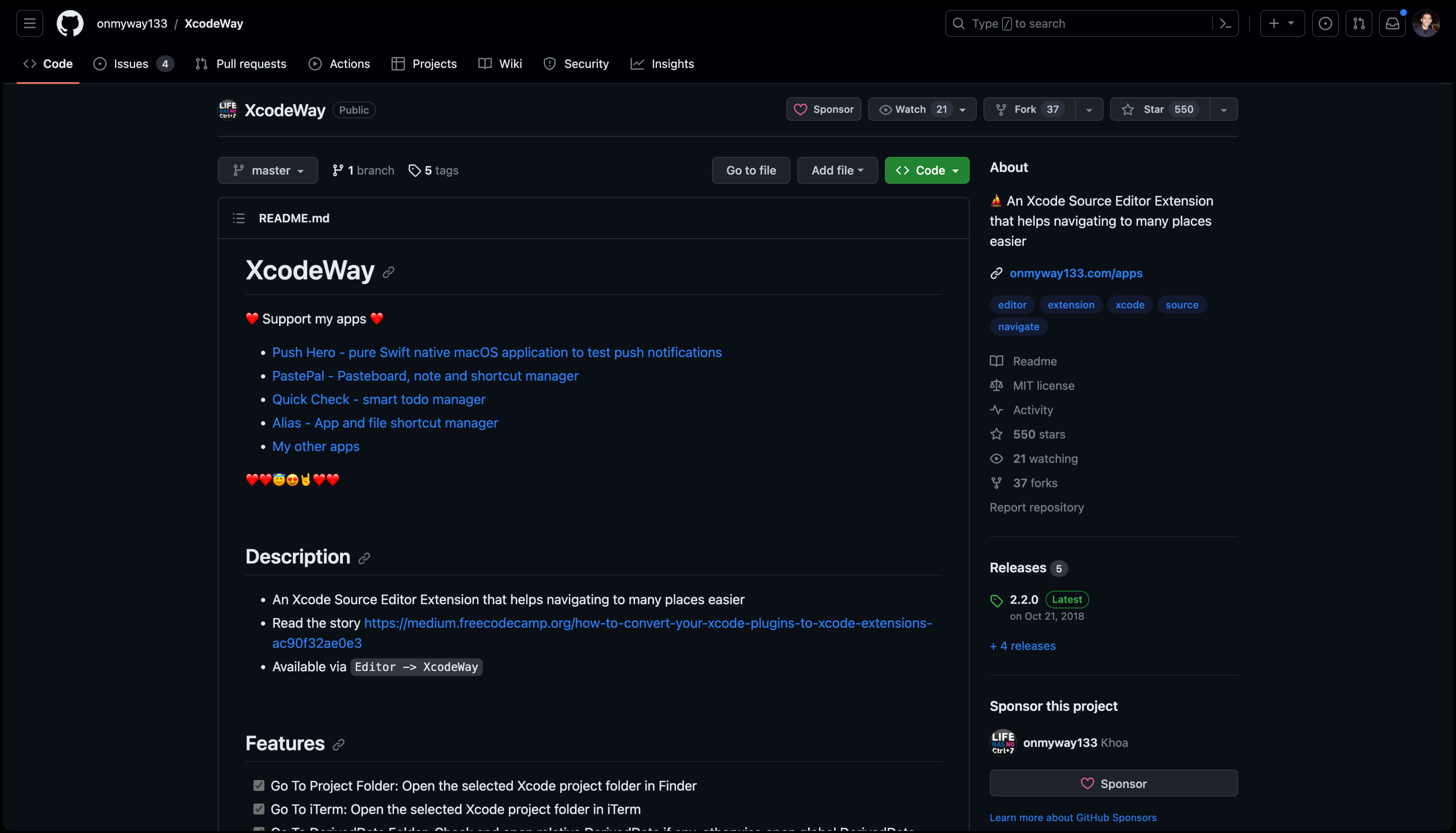
```swift
class AppleScriptRunner {
    static func run() {
        // Get the URL for the AppleScript file
        guard let applicationDirectoryPath = try
                   FileManager.default.url(for: .applicationScriptsDirectory,
                   in: .userDomainMask, appropriateFor: nil, create: true) else {
            return
        }

        let scriptPath = applicationDirectoryPath.appendingPathComponent("SwiftLeeds.scpt")

        // Check if the AppleScript file exists and create an NSUserAppleScriptTask
        guard FileManager.default.fileExists(atPath: scriptPath.path),
                let script = try? NSUserAppleScriptTask(url: scriptPath) else {
            return
        }

        // Execute the AppleScript with a nil event (default subroutine)
        script.execute(withAppleEvent: nil) {_, error in
            if let error = error {
                print(error)
            }
        }
    }
}
```

Khoa Pham

@onmyway133

Boris Bügling

@NeoNacho



XcodeWay

❤️ Support my apps ❤️

- Push Hero - pure Swift native macOS application to test push notifications
- PastePal - Pasteboard, note and shortcut manager
- Quick Check - smart todo manager
- Alias - App and file shortcut manager
- My other apps

❤️❤️😇😍🤘❤️❤️

Description

- An Xcode Source Editor Extension that helps navigating to many places easier
- Read the story https://medium.freecodecamp.org/how-to-convert-your-xcode-plugins-to-xcode-extensions-ac90f32ae0e3
- Available via `Editor -> XcodeWay`

Features

- [x] Go To Project Folder: Open the selected Xcode project folder in Finder
- [x] Go To iTerm: Open the selected Xcode project folder in iTerm

About

⛵ An Xcode Source Editor Extension that helps navigating to many places easier

onmyway133.com/apps

editor extension xcode source navigate

Readme
MIT license
Activity
550 stars
21 watching
37 forks
Report repository

Releases 5

2.2.0 Latest
on Oct 21, 2018

+ 4 releases

Sponsor this project

onmyway133 Khoa

Sponsor

Learn more about GitHub Sponsors

What We’ll Cover

- 📚 Overview
- 🚀 Building Editor Extensions
- 💥 Breaking The Rules
- 📦 **Distribution**
- 📥 Installation

Sharing Your Extension

- Editor Extensions require a **hosting macOS app**
- **Store preferences** within the host app
- **Host app should contain UI**; extensions can't have their own
- Distribute via the **Mac App Store**
- Distribute via your own **Developer ID**

theswiftdev / awesome-xcode-extensions
Type / to search
Code
Pull requests 1
Actions
Security
Insights
awesome-xcode-extensions Public
Watch 112
Fork 216
Star 3k
master
1 branch
0 tags
Go to file
Add file
Code
README.md

Awesome native Xcode extensions

Awesome native Xcode extensions. Feel free to contribute!

Contributing

Please submit a pull request to improve this file. Thank you to all contributors; you rock!

The list

Tutorials of Xcode Source Editor Extension

- XTExtension - Comment lines.
- XcodeExtensionSample - Various sample commands for your Xcode Source Editor Extension implementation.
- Xcode Source Editor Extension Tutorial: Getting Started

Formatters

- 🔮 Snowonder — an import declarations formatter Xcode Extension.
- Imp😈 - Sorting imports in Xcode files has never been that fun and easy
- XAlign - An amazing Xcode Source Editor extension to align regular code.
- Alignment -This Xcode source editor extension align your assignment statement.
- CleanClosureXcode - An Xcode Source Editor extension to clean the closure syntax.

About

Awesome native Xcode extensions.

theswiftdev.com/2017/10/05/awesome...

plugin awesome extension xcode xcode-plugin xcode-extension source-editor-extension

Readme
Activity
3k stars
112 watching
216 forks
Report repository

Releases

No releases published

Packages

No packages published

Contributors 66

aryamansharda / EditKitPro
Type / to search
Code
Issues
Pull requests
Actions
Projects
Security
Insights
Settings
EditKitPro
Public
Pin
Unwatch 6
Fork 6
Starred 96
main
1 branch
0 tags
Go to file
Add file
Code
README.md
EDIT KIT PRO
SUPER-CHARGE YOUR IOS DEVELOPMENT WORKFLOW

About

A multi-purpose Xcode Editor Extension for iOS and macOS developers

digitalbunker.dev/editkit-pro/

editor mac apple extension xcode

- Readme
- MIT license
- Activity
- 96 stars
- 6 watching
- 6 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

EditKitPro

EditKit Pro provides a suite of tools to help you write better, cleaner, and more efficient code. Whether you need to quickly format your code, create Codable models, generate mock data, or move around in SwiftUI more efficiently, EditKit Pro has you covered.

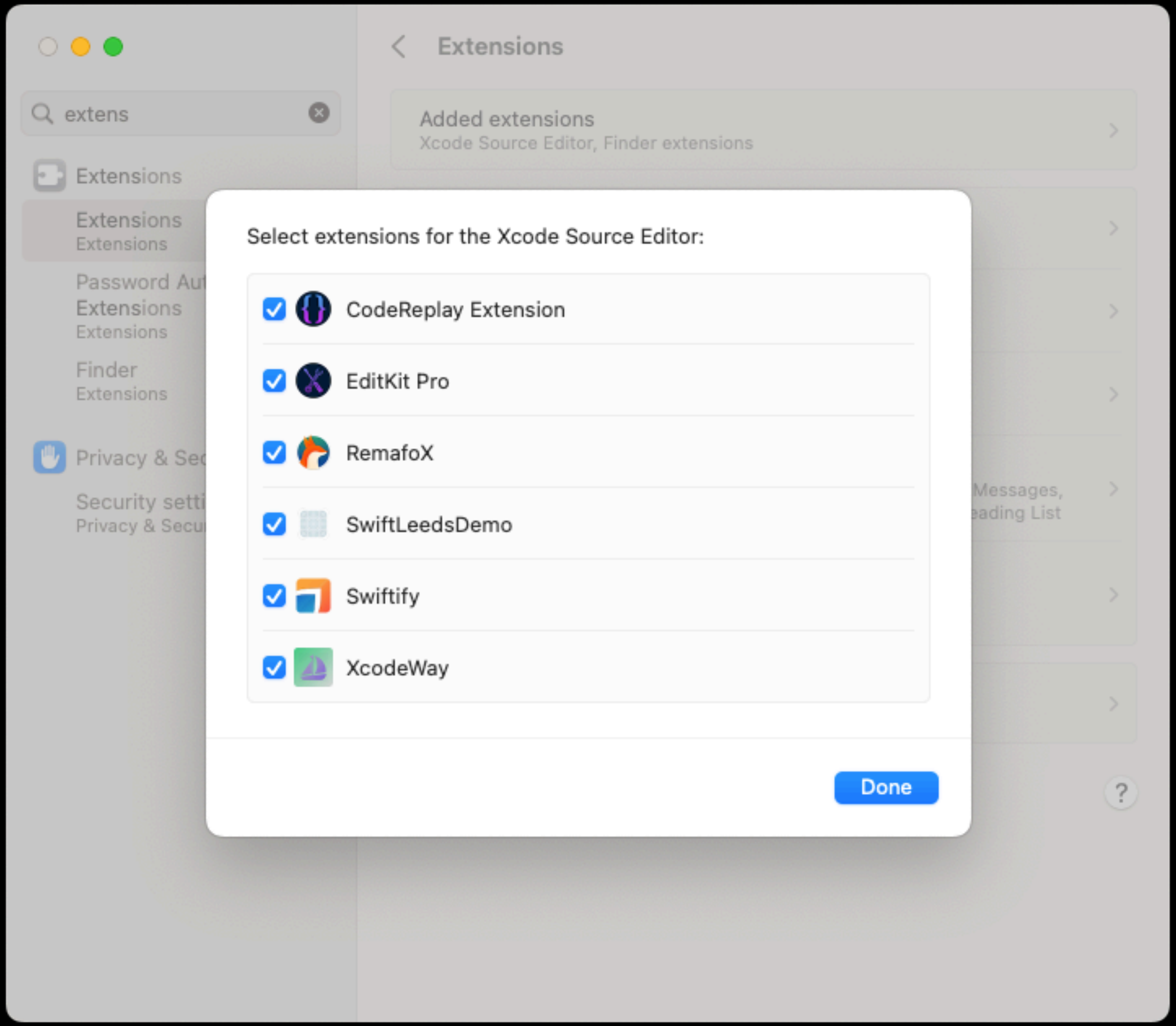
This is an open-source Xcode Editor Extension with a variety of mini-tools for iOS / macOS Developers.

Demos of EditKit can be found on the blog post and this YouTube Video.

What We'll Cover

- 📚 Overview
- 🚀 Building Editor Extensions
- 💥 Breaking The Rules
- 📦 Distribution
- 📥 **Installation**

Launch



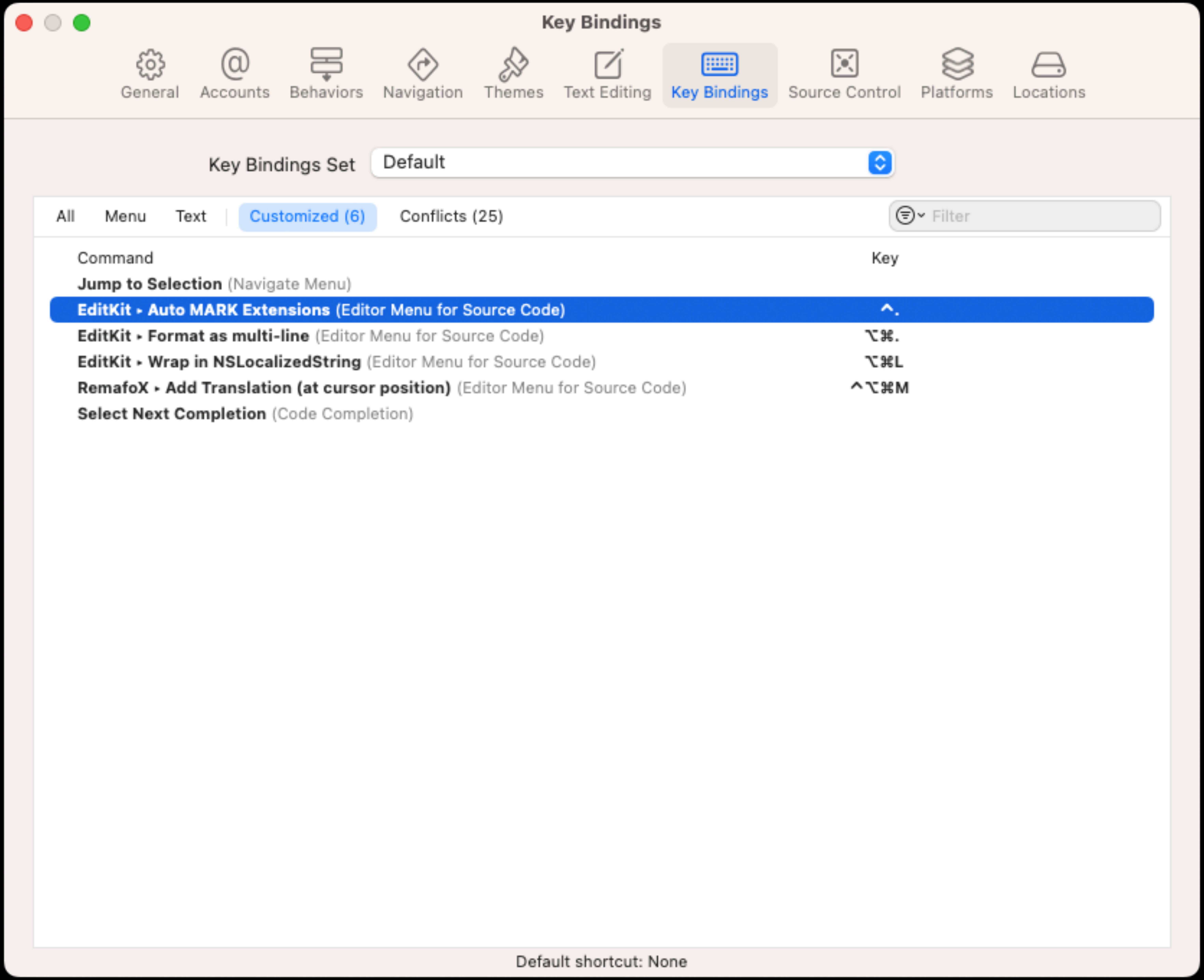
Added extensions
Xcode Source Editor, Finder extensions

Select extensions for the Xcode Source Editor:

- ✓ CodeReplay Extension
- ✓ EditKit Pro
- ✓ RemafoX
- ✓ SwiftLeedsDemo
- ✓ Swiftify
- ✓ XcodeWay

Restart

Enable in System Preferences



Key Bindings Set: Default

Command	Key
Jump to Selection (Navigate Menu)	
EditKit ▸ Auto MARK Extensions (Editor Menu for Source Code)	^.
EditKit ▸ Format as multi-line (Editor Menu for Source Code)	⌥⌘.
EditKit ▸ Wrap in NSLocalizedString (Editor Menu for Source Code)	⌥⌘L
RemafoX ▸ Add Translation (at cursor position) (Editor Menu for Source Code)	^⌥⌘M
Select Next Completion (Code Completion)	

Default shortcut: None

Setting up Key Bindings

✅

Recap

- 📚 Overview
- 🚀 Building Editor Extensions
- 💥 Breaking The Rules
- 📦 Distribution
- 📥 Installation

digitalbunker.

Slides & Code

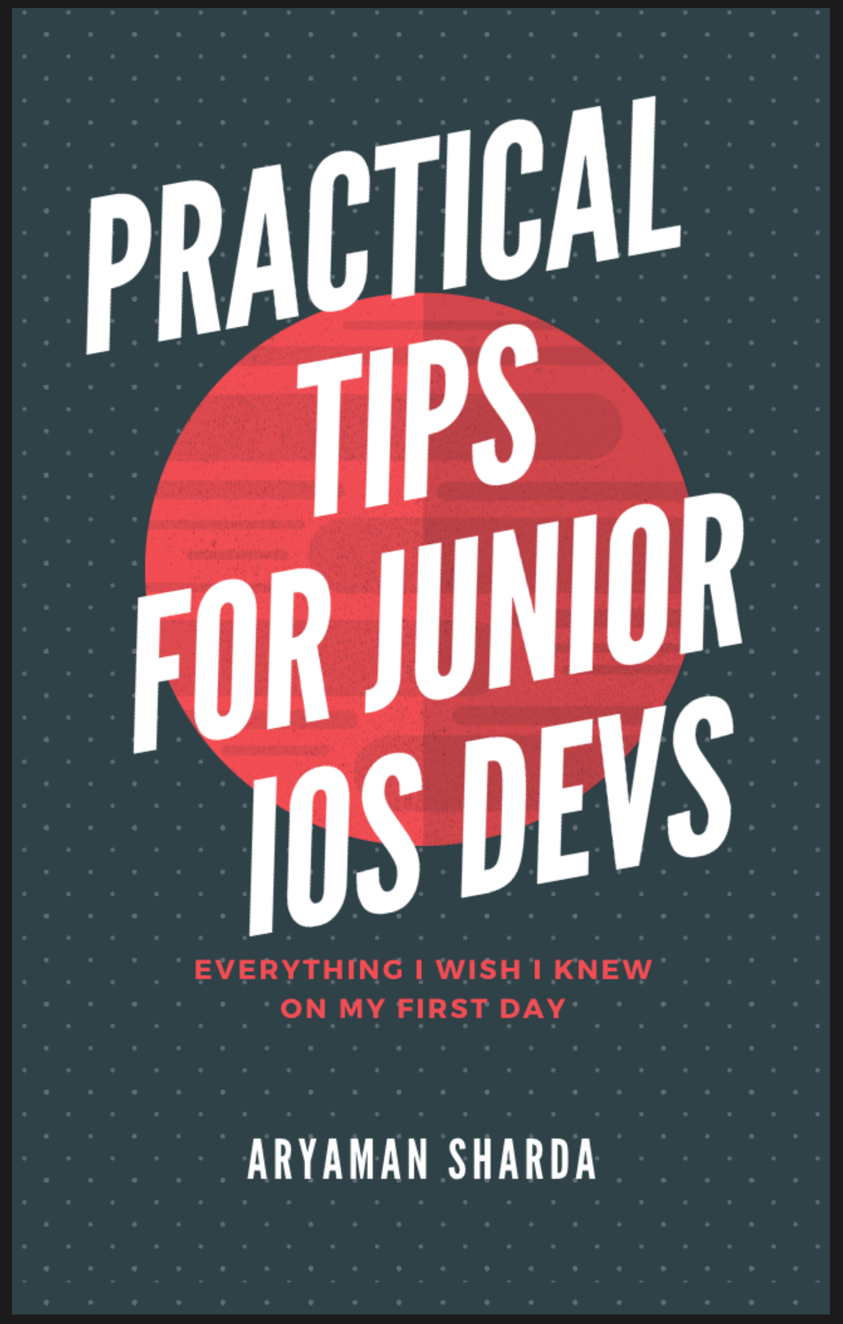
Aryaman Sharda

👋 Thanks SwiftLeeds!

✍️ Blog at **digitalbunker.dev**

✉️ Curator of the **indie.watch** newsletter

@aryamansharda


PRACTICAL TIPS FOR JUNIOR IOS DEVS
EVERYTHING I WISH I KNEW ON MY FIRST DAY
ARYAMAN SHARDA

slido.com
#SL23

SwiftLeeds